AVDS

User's Manual

AVIATOR VISUAL DESIGN SIMULATOR (AVDS)

User Manual

(Windows NT/95/98/2000/XP)

February 2006

RASMUSSEN SIMULATION TECHNOLOGIES, LTD.

SOFTWARE LICENSE

IMPORTANT: PLEASE READ THIS SOFTWARE LICENSE AGREEMENT ("LICENSE") CAREFULLY BEFORE OPENING THIS PACKAGE OR INSTALLING THIS SOFTWARE. BY OPENING THE PACKAGE OR INSTALLING THIS SOFTWARE YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS LICENSE. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENSE, RETURN THIS PRODUCT TO RASMUSSEN SIMULATION TECHNOLOGIES, LTD.

1. License. The software ("Software") and documentation whether on disk, in read only memory, on any other media or in any other form are licensed to you by Rasmussen Simulation Technologies, Ltd. ("RasSimTech") for internal purposes only and to make one copy of the Software in machine readable form for back-up purposes. You must put the same copyright and other proprietary rights notices on any copy of the Software as it appears on the original. You own the media on which the RasSimTech Software is recorded but RasSimTech and/or RasSimTech's licensor(s) retain title to the RasSimTech Software. The Software may be used only on a single computer owned, leased, or otherwise controlled by you; or in the event of the inoperability of that computer, on a backup computer selected by you. Neither concurrent use on two or more computers nor use in a local area network or other network is permitted without Licensor's further authorization. The RasSimTech Software and any copies made and/or distributed under this License are subject to this License.

2. Permitted Uses and Restrictions. Except as permitted by applicable law and this License, you may not decompile, reverse engineer, disassemble, modify, rent, lease, loan, distribute, create derivative works from the Software or transmit the Software over a network. The Software may not be sold, leased, assigned, sublicensed, or otherwise transferred in whole or in part. You may not distribute, modify, adopt, transfer, or create derivative works of the documentation or master Software.

3. Termination. The License is effective until terminated. You may terminate this agreement at any time by destroying the Software and all copies thereof. Your rights under this License will terminate automatically without notice from RasSimTech if you fail to comply with any term(s) of this License. Upon termination you must destroy the Software and all copies thereof.

4. Disclaimer of Warranty. You expressly acknowledge and agree that use of the Software is at your sole risk. The Software is provided AS IS and without warranty of any kind and RasSimTech and RasSimTech's licensor(s) (for the purposes of provisions 4 and 5, RasSimTech and RasSimTech's licensor(s) shall be collectively referred to as "RasSimTech") EXPRESSLY DISCLAIM ALL WARRANTIES AND/OR CONDITIONS, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES AND/OR CONDITIONS OF MERCHANTABILITY OR SATISFACTORY QUALITY AND FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. RasSimTech DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE SOFTWARE WILL MEET YOUR REQUIREMENTS, OR THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT DEFECTS IN THE SOFTWARE WILL BE CORRECTED. FURTHERMORE, DOES NOT WARRANT OR MAKE ANY

REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF THE SOFTWARE OR RELATED DOCUMENTATION IN TERMS OF THEIR CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY OR AN AUTHORIZED REPRESENTATIVE SHALL CREATE A WARRANTY OR IN ANY WAY INCREASE THE SCOPE OF THIS WARRANTY. SHOULD THE SOFTWARE PROVE DEFECTIVE, YOU (AND NOT OR AN AUTHORIZED REPRESENTATIVE) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THE TERMS OF THIS DISCLAIMER DO NOT AFFECT OR PREJUDICE THE STATUTORY RIGHTS OF A CONSUMER ACQUIRING RasSimTech PRODUCTS OTHERWISE THAN IN THE COURSE OF A BUSINESS, NEITHER DO THEY LIMIT OR EXCLUDE ANY LIABILITY FOR DEATH OR PERSONAL INJURY CAUSED BY RasSimTech's NEGLIGENCE.

5. Limitation of Liability. UNDER NO CIRCUMSTANCES, INCLUDING NEGLIGENCE, SHALL RasSimTech BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO THIS LICENSE. SOME JURISDICTIONS DO NOT ALLOW THE LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THIS LIMITATION MAY NOT APPLY TO YOU. In no event shall RasSimTech's total liability to you for all damages exceed the amount of fifty dollars ($50.00).

6. Export Regulations. You may not use or otherwise export or reexport the Software except as authorized by United States law and the laws of the jurisdiction in which the Software was obtained. In particular, but without limitation, the Software may not be exported or reexported (i) into (or to a national or resident of) any U.S. embargoed country or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Department of Commerce's Table of Denial Orders. By using the Software, you represent and warrant that you are not located in, under control of, or a national or resident of any such country or on any such list.

7. Government End Users. If the Software is supplied to the United States Government, the Software is classified as "restricted computer software" as defined in clause 52.227-19 of the FAR. The United States Government's rights to the Software are as provided in clause 52.227-19 of the FAR.

8. Controlling Law and Severability. This License is made under, shall be governed by and construed in accordance with the laws of the United States and the State of Ohio, irrespective of its choice of law provisions.

9. Complete Agreement. This License constitutes the entire agreement between the parties with respect to the use of the Software and supersedes all prior or contemporaneous understandings regarding such subject matter. No amendment to or modification of this License will be binding unless in writing and signed by RasSimTech .

Table of Contents

Chapter 1 Introduction

1.1 Description

The Aviator Visual Design Simulator (AVDS) is a set of PC-based flight simulator and visualization tools designed to be used by engineers, students, educators, and other researchers interested in aerospace research and development.

1.1.1 Users and Uses

The following list describes potential users and the functions and capabilities provided by AVDS:

- *Beginning Aeronautical/Aerospace Engineering Students* - to explore and solve aeronautical/aerospace problems set up by the instructor.
- *Aero Educators* - to demonstrate aeronautical principles to students.
- *Advanced Aero Students and their Facility Advisors* - for advanced aeronautical research.
- *Aircraft Designers* - to interactively simulate designs and animate the result of batch simulations.
- *Prototype/Concept Aircraft Designers* - to animate mock-up aerospace vehicles for demonstration and marketing.
- *Flight Control System Designers* - for flight control system research and development.
- *Aircraft Component Designers* (Engines, Inlets, Wings, INS/GPS, etc.) - to simulate the component on a surrogate aircraft over the operating envelope.
- *Flight Test Engineers* - to fly the flight test maneuvers before the actual test, download, and animate flight test data real-time and animated post-test analysis.
- *Accident Reconstruction* - to reconstruct accidents from black box data to find cause and to help prevent future accidents.

1.1.2 User Prerequisites

Designed as a joint project between Artificial Horizons, Inc., (AHI) and the United States Air Force, the software and manual were created based on the user already having a certain level of understanding and knowledge of basic flight principles, and aircraft operation.

- On a basic level, the user only needs to know how to log into their account and run the program.
- On a more advanced level (e.g., for Interactive Simulation), the user needs to have a working knowledge of C programming (and other computer languages, if needed), and aircraft/vehicle simulation.

1.2 System Requirements

The following system specifications are required to install and run the AVDS Version 1.3:

Minimum

- Microsoft® Windows 95/98/NT/2000/XP or later
- 32 MB of RAM
- At least 120 Megabytes (MB) of disk space

Recommended

- Microsoft® Windows 95/98/NT/2000/XP or later
- A Graphics Accelerator capable of hardware acceleration of OpenGL(TM)
- 128 MB of RAM
- At least 250 Megabytes (MB) of disk space

1.3 Features

AVDS provides many highly sophisticated features as shown in the following list:

- Distribute tools over many computers on network
- Distributed interactive simulation
- Highly detailed, USGS compatible terrain
- Two-user definable, interactive simulation models, nominally Flight Control System (FCS) and aircraft
- User-definable, 3-dimensional parameter look-up tables
- User-defined interconnections between simulations
- Feedback of simulation parameters, such as pitch rate, altitude, airspeed, aircraft latitude, weight on main landing gear, target lock on, weapon select, weapons release, etc.
- Interactive simulation with external hardware inputs
- Interactive, hardware in the loop simulation capability
- User definable aircraft images
- User definable, articulated surfaces (i.e., surfaces that rotate based on the simulation.)
- User can link in existing simulation code, can distribute simulation by running dynamics code on one computer and AVDS on another, can make batch simulation run interactively, can run interactive simulation on one computer and control its operation on another, and can run one craft interactive while animating others (i.e., they can interact).

1.4 Version

This version contains all of the tools necessary to give the aerospace researcher a better understanding of a situation or problem and information that may be useful in diagnosing and problem solving.

1.5 Feedback

RasSimTech Ltd, appreciates any feedback regarding this first version of the AVDS. Please contact us by electronic mail at:

avds-feedback@RasSimTech.com.

1.6 Technical Support

For technical support, you may reach us by electronic mail at: **avds-techsup@RasSimTech.com** or by Voice Mail at ***614.487.1056***.

1.7 Problem Resolution

If the latest release of software exhibits a problem, then depending on the severity of the problem, one of two steps will be taken:

- An immediate resolution will be found, or
- A resolution will be incorporated into the next release of software.

Chapter 2 Quick Reference

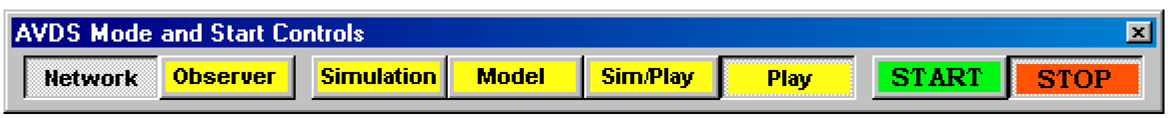
2.1 Toolbars

Avds uses dockable tool bars, which means that the user can re-positioned or un-dock any of the toolbars to a separate window. To move a toolbar, select the border of the bar and, while holding the left mouse button down, move the toolbar to the desired position on the AVDS frame window. To un-dock a tool bar hold down the **Ctrl** and then click on a border of the toolbar using the left mouse button.

The toolbars are controlled from the **Toolbars menu**, shown below.

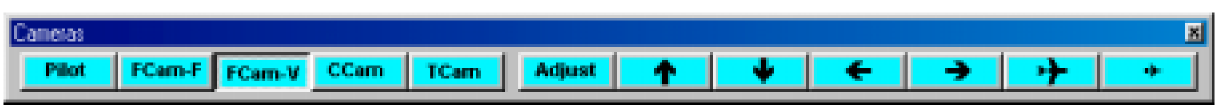
As shown in the menu the toolbars are: **Main**, **Cameras**, **Indicators**, and **Playback**. Also included in this menu are controls for turning the **Message** window and the **Status** bar off/on.

2.1.1 Main Toolbar

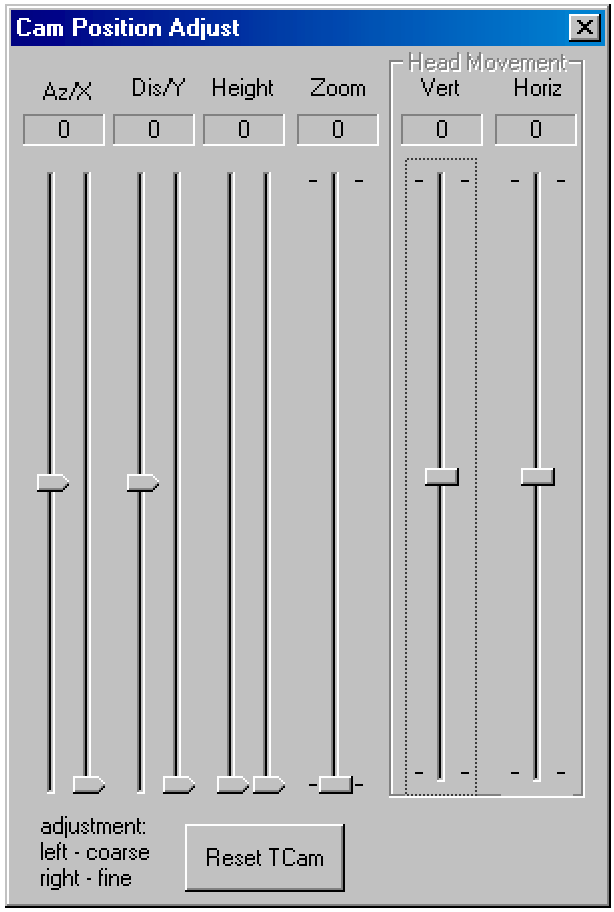


The main toolbar contains a button to switch **Network** mode on and off and other buttons to control the AVDS modes of operation: **Observer**, **Simulation**, **Model**, Simulation and Playback (**Sim/Play**), and Playback (**Play**). This toolbar also contains buttons for beginning and ending, **START** and **STOP**, AVDS operation in the chosen mode.

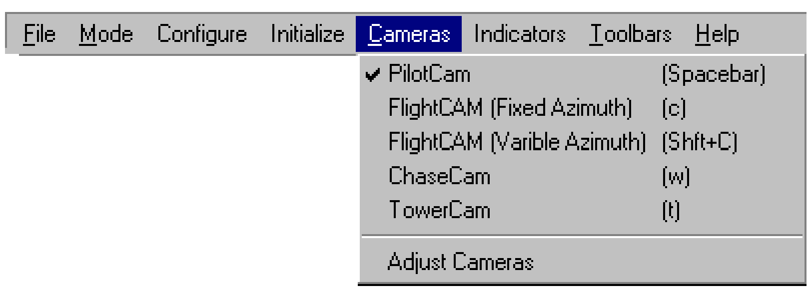
2.1.2 Cameras Toolbar



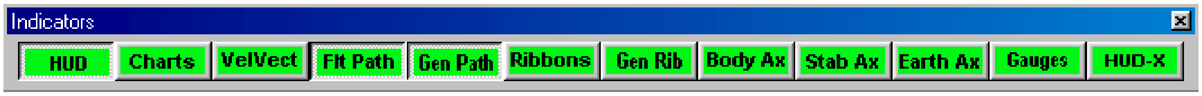
The **Camera toolbar** contains buttons to select the current camera mode, point-of-view and make adjustment to the camera position. There are five camera modes **Pilot**, **FCam-F** (FlightCam with fixed azimuth), **FCam-V** (FlightCam with variable azimuth), **CCam** (ChaseCam), and **TCam** (TowerCam). The **Adjust** button brings up the **Camera Position Adjustment window,** which enables the user to adjust camera position using sliders, shown below.



The rest of the buttons on the **Camera bar** adjust the position of the camera, based on the type of camera. The and buttons change the cameras vertical position. The and buttons change the cameras horizontal position. The and buttons move the cameras closer or farther away from the aircraft. Note: in the case of the TCam the and buttons change the zoom factor. Many of the commands on the **Cameras toolbar** can also be found in the **Cameras menu** as shown below.

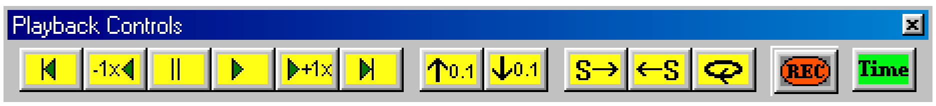


2.1.3 Indicators Toolbar

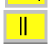
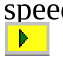
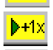
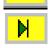
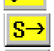
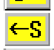


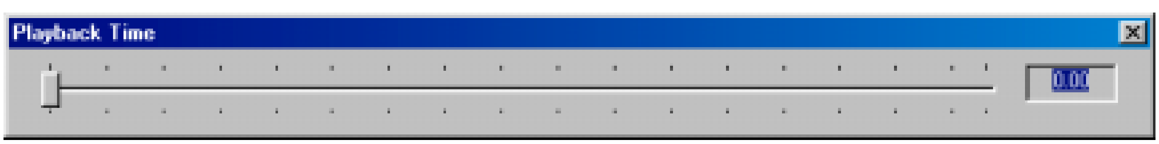
The **Indicators toolbar** contains buttons that toggle the various indicators on and off. The indicators are: **HUD**, Strip Charts (**Charts**), Velocity Vector (**VelVect**), Flight Path Marker display (**Flt Path**) and generate (**Gen Path**), Wing-Tip Ribbons display (**Ribbons**) and generate (**Gen Rib**), Body Axes (**Body Ax**), Stability Axes (**Stab Ax**), Earth oriented axes (**Earth Ax**), simulation gauges (**Gauges**), and display of cross hairs in the HUD (**HUD-X**).

2.1.4 Playback Toolbar

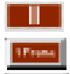


The Playback toolbar contains buttons that control **playback** and **recording**. The functions are:

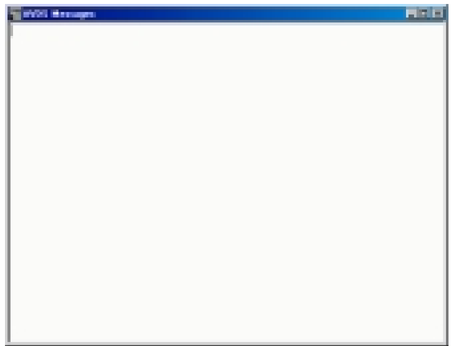
 - **rewind** to the beginning of the data (set time=0)
 - **decrease speed multiplier by one**
 - **pause/resume** (pause sets speed to 0, resume sets speed to last speed)
 - **forward one speed** (sets speed to 1)
 - **increase speed multiplier by one**
 - **Fast forward** (set time to maximum time of the data)
 - **increase speed multiplier by 0.1**
 - **decrease speed multiplier by 0.1**
 - **single time step forward**
 - **single time step backward**
 - **continuous loop** (at the end of the data time is reset to 0)
 - **record** simulation data
- display "**Playback Time**" toolbar



2.1.5 Video Toolbar

 - initialize AVDS for recording video
 - pause/upause video recording
 - record one frame of video
 - record one second of video

2.2 Message Window



AVDS prints all notice, warning, and error messages to the **Message window**. The message area is scrollable so all messages can be reviewed. Messages are not saved to a file so all messages generated while running AVDS are discarded when the user exits. The Message window can be hidden by pressing the **X** in the top right corner of the **Message window** or by selecting **Messages** in the **Toolbars menu**.

2.3 Status Bar

The **Status Bar** shows helpful messages about the user interface as well as the status of AVDS. The **Status bar** is not dockable and is fixed to the bottom of the main AVDS window. The first half of the **Status bar** displays tips about the AVDS control that the mouse is currently over. That is, the user can gain information about the workings of a button or other control by positioning the mouse over the control and reading the message in the **Status bar**. Selecting **Status Bar** in the Toolbars menu will hide the Status bar. There are also five other message windows on the status bar. The first displays the mode the use of the rest varies based on the mode of operation:

MODE	Window 2	Window 3	Window 4	Window 5
Observer	Number of simulations on the network	Type of currently selected simulation	Handle of currently selected simulation	Graphics frame update rate in Hertz
Simulation, Model	N/A	If a joystick is attached, input sample rate of the joystick in Hz	Maximum update rate of the dynamic simulations	Graphics frame update rate in Hz
Sim/Play, Play	Current playback speed multiplier	Number of current vehicle selected	N/A	Graphics frame update rate in Hertz

2.4 Control Commands

The following tables are a quick reference to the commands used in AVDS. These commands are described in more detail in later chapters.

Table 2-1 Keyboard Controls for All Modes

Command Key	Description
Start	Begins simulation or playback.
Stop	Ends simulation or playback.
Space	Switch view to other vehicles
Shift + space	Switch view to primary vehicle
0, 1, 2, 3, 4, 5, 6, 7, 8, 9	Switch to selected vehicle number. *Note*: *'0' is the primary simulation vehicle*.
F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12	Holding down one of these keys while selecting one of the 0-9 keys adds an offset to the selected vehicle number. F2 adds 10, F3 adds 20, F4 adds 30, … and F12 adds 110. For example, to select vehicle 52, press the F6 and 2 keys at the same time.
i	Flight path marker display on/off
I	Flight path marker generation on/off
m	Wing tip ribbon display on/off
M	Wing tip ribbon generation on/off
#	Strip charts on/off
y	Velocity vector on/off
e	Earth axes on/off
x	Stability axes on/off
n	Body axes on/off
h	HUD off/on

Table 2-2 Mouse Control for Interactive Simulation and Networked Modes (with Joystick enabled)

Mouse Button	Resulting Action
Left	Move mouse forward/back to set throttle*
Both	Changes set point of camera during flight-cam view
Right	Move mouse left/right to change rudder**
Forward/Back/Left/Right	Rotates viewing perspective

*This command is active when the Joystick Configuration check box **Use Keyboard/Mouse for Throttle** is checked.

** *This command is active when the Joystick Configuration check box* **`Use Keyboard/Mouse for rudder`** *is checked.*

Table 2-3 Keyboard Commands for Camera Views, Available in All Modes

Command	Action	Description

w	Wingman view	Flies in formation with aircraft
c	Flight-cam view	Tied to aircraft center of gravity, heading set with respect to the earth
C	Flight-cam view	Tied to aircraft center of gravity, heading rotates with the aircraft
t	Track-cam view	Stationary position, rotates to follow aircraft
T	Track-cam view	Changes track camera position to present aircraft location
[or {	Moves closer	Moves camera toward aircraft
] or }	Moves away	Moves camera away from aircraft
.. or >	Pan Right	Shift heading counter-clockwise about the aircraft
, or <	Pan Left	Shift heading clockwise about the aircraft
= or +	Pan Up	Moves camera up
- or _	Pan Down	Moves camera down
Z/z	Zoom In/Out	Zoom track camera in/out

Table 2-4 Mouse Control for Flight-Cam View, All Modes

Mouse Button	Resulting Action
Both Buttons	Pressing these buttons with motion of the mouse changes the set point of the flight-cam view

Table 2-5 Keyboard Commands for Interactive Simulation Modes

Command	Resulting Action
Joystick select or s	Select weapon
S	Select no weapon
Joystick fire or Return	Fire selected weapon
Joystick throttle or a	Increase throttle
Joystick throttle A	Maximum throttle
Joystick throttle d	Decrease throttle
Joystick throttle D	Minimum Throttle
b	Brakes on/off
f	Flaps down
F	Flaps up
^	Start recording flight data (erases previously saved data, in memory, if it exists
g	Gear up/down

Table 2-6 Mouse Control for Interactive Simulation and Networked Modes (without Joystick enabled)

Mouse Movement	Resulting Action
Forward/Backward	Commands stick pitch inputs
Left/Right	Commands stick roll inputs
Left Button + Forward/Backward	Changes throttle value

Both Buttons + Movement	Changes set point of camera during flight-cam view
Right Button + Left/Right	Command rudder

Table 2-7 Numeric Keypad Commands for Playback Mode

Keypad Command	Resulting Action
6 (Left Arrow)	Increase forward speed by 1X
4 (Right Arrow)	Decrease forward speed by 1X
5	Pause
9 (Pg Up)	Forward normal speed
8 (Up Arrow)	Increase speed scale factor by 0.1
2 (Down Arrow)	Decrease speed scale factor by 0.1
7 (Home)	Beginning of playback file
1 (End)	End of playback file

Chapter 3 Running AVDS

3.1 AVDS Set-Up

When AVDS is installed, there are eight main subdirectories under the installation directory. They are **bin**, **craft**, **manual**, **model**, **ports**, **terrain**, **userfiles,** and **Utilities**, see **Figure 3-1**. The **Utilities** directory contains five main subdirectories**: Utilities/GeometryUtility, Utilities/MATLABToolbox, Utilities/network, Utilities/SharedMemoryLibray,** and **Utilities/terrain**. The content of each of these is described below. The directories that are most important to the user are **craft**, **model** and **userfiles**. These three directories contain files that users modify to perform their desired interactive simulation or data visualization. Before modifying files in these directories, it is recommended that the user make a copy of the directories and use the copied directories as working directories. To use non-default directories, the user must modify the **avds.ini** file, with either a text editor or in the **properties** window in AVDS, see Section 3.4.1.4 below.

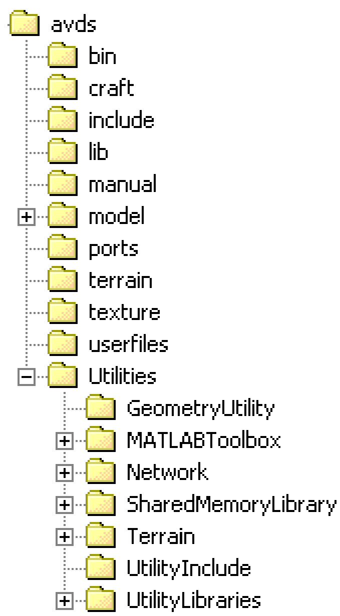


Figure 3-1 AVDS Directory Structure

3.1.1 The bin directory:

The **bin** directory contains three files, **avds.exe** and **avds.ini**. This is the directory that AVDS should be started from. This can be done by either double-clicking the **avds.exe** icon or by selecting AVDS from the **windows Start menu**. The **avds.ini** file is a text file that is in standard windows ini format that contains file and path names for the AVDS setup file and the AVDS directories. When the AVDS program is initiated, it reads the ***avds.ini*** *file*. If this file does not exist, AVDS builds a new ***avds.ini*** file based on the installation directory and assumes the default directory structure **Figure 3-1**.

When users make working copies of AVDS directories, they must update the avds.ini file either with a text editor, or (recommended) by modifying the entries in the properties window, **Figure 3-4**.

3.1.2 The craft directory:

The **craft** directory contains files that enable the user to change the appearance of existing, or add new aircraft images. Files in this directory include a **craftcap** file used to identify aircraft image definition files, a compiled version of this file, **craftcap.o**, and sample aircraft image definition files. See **Appendix A**, Craftcap file formats, for more information on the craftcap and aircraft image definition files.

3.1.3 The manual directory:

The **manual** directory contains this manual and other documentation.

3.1.4 The model directory:

The **model** directory contains files that enable the user to change the structure of existing, or add new FCS and aircraft simulations. Files in this directory, and subdirectories include a **modelcap** file used to identify user defined models, sample MS Visual C++ projects used to compile user code into a suitable dynamic link library, and sample FCS and aircraft models source code. See **Chapter 4** for information on adding **user defined models** to AVDS.

3.1.5 The ports directory:

The **ports** directory contains files that define the airport and other structural images used in AVDS. Files in this directory include a **portscap** file used to identify airport image definition files, the airport image definition files, and compiled airport files. See **Appendix A**, Portscap file formats, for more information on the **portscap** and airport image definition files.

3.1.6 The terrain directory:

The **terrain** directory contains terrain data files used in AVDS. AVDS derives the latitude and longitude of the terrain block from the name of the file. The convention for naming these files is XXX_YYY, where:

XXX = **((longitude +180.0) * 1.171875) + ((longitude +180.0) * 9.375) & 0x01);**

YYY = **((latitude+90.0) * 1.171875) + ((latitude+90.0) * 9.375) & 0x01)**

3.1.7 The userfiles directory:

The **userfiles** directory contains files that configure the simulation or data playback. The files include:

***.sim.ini, *.ply.ini** - These are sample configuration files for the Aircraft Initialization Window and the Playback Configuration Window, respectively.

These files are used to save configurations and can be copied and customized by the user.

file.cam - This is a sample file containing camera commands used during playback, see **Section 5.2.2.10**.

***.av.txt, *.fcs.txt** - These are example parameter look-up table files. These files are used to modify simulation dynamics and can be changed between simulation runs to change aircraft and flight control system simulation dynamics for the Aviator Aircraft and the Feedback Gains flight control system, see **Section 4.6.2**.

RecordData.save.txt - This is the sample playback data file.

AVDS_Global.ini - This file is the **Global Setup File**, used to save all of the default settings for AVDS, except directories. The file is created/saved when changes are made in AVDS.

3.1.8 The utilities/GeometryUtility directory:

The **utilities/GeometryUtility** directory contains a utility to aid in conversion from AutoCAD's DXF format to AVDS craft format. For more information see **Appendix E**.

3.1.9 The utilities/MATLABToolbox directory:

The **utilities/MATLABToolbox** directory contains MATLAB®, Simulink®, and AVDS utilities to make connections between AVDS and MATLAB/Simulink. For more information see the **AVDS – MATLAB Toolbox Reference Manual** which can be found in the directory "*AVDS\Utilities\MATLABToolbox*".

3.1.10 The utilities/network directory:

The **utilities/network** directory contains sample code and libraries for developing external simulations that are networked with AVDS. For more information see **Appendix B**.

3.1.11 The utilities/SharedLibray directory:

The **utilities/SharedLibray** directory contains the source code used to build the shared memory library. This library can be used to make connections between programs using shared memory.

3.1.12 The utilities/terrain directory:

The **utilities/terrain** directory contains subdirectories that contain sample code and libraries for converting files containing elevation, and color, data into AVDS terrain data files. For more information see **Appendix D**.

3.2 User Customization

Users can customize AVDS by changing parameters in parameter look-up tables, creating or modifying simulations, creating or modifying aircraft

images, changing airports or adding ground structures, adding new terrain, and developing and implementing networked simulations. To customize AVDS, the user should copy the appropriate AVDS directory to a local directory. Available directories are the following:

`userfiles` - Use this directory for changing parameters in parameter look-up tables. See **Section 4.6**, *Aircraft Model And Flight Control System Modifications.*

`model` - Use this directory for creating or modifying simulations. See **Section 4.6**, *Aircraft Model And Flight Control System Modifications*.

`craft` - Use this directory for creating or modifying aircraft images. See the *Craftcap Man Pages* in **Appendix A**.

`ports` - Use this directory for changing airports or adding ground structures. See the *Portscap Man Pages* in **Appendix A**.

`utilities/terrain` - Use this directory for adding new terrain. See the *AVDS Terrain Library Functions* in **Appendix D**.

`utilities/network` - Use this file for developing and implementing networked simulations. See the *AVDS Network Interface Library Functions* in **Appendix B**.

`utilities/` GeometryUtility - Use this file for coveting #D files into AVDS format.. See *DXF to AVDS Conversion Utility* in **Appendix E**.

3.3 AVDS Main Window

AVDS begins by opening the main window, **Figure 3-2**. This window gives the user the ability to choose and modify the mode of operation. From the Main window, there are setup menus and windows to configure and further refine Interactive Simulation or Playback. The setup windows and menus are described below.

Figure 3-2 AVDS Main Window

3.4 Setup menus and Windows

3.4.1 File Menu

The file menu contains sub-menus for loading and saving global initialization files, simulation initialization files and playback initialization files, see **Figure 3-3**. This menu also contains the command, **Properties...**, to open the Properties window. Each of these is described below.

3.4.1.1 File → Global Setup

The global setup file. The default Global Setup file is **AVDS_Global.ini. See Appendix A** for more information on the format of global setup file.

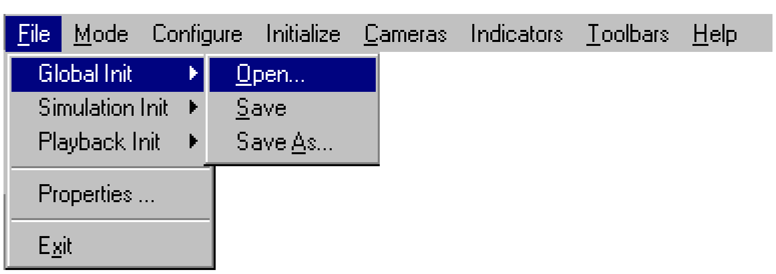


Figure 3-3 File menu

3.4.1.2 File → Simulation Init

The simulation configuration file. The default file extension is ***sim.ini***. Refer to **Chapter 4** on *Interactive Simulation* for more information on the simulation configuration file.

3.4.1.3 File → Playback Init

The playback configuration file. The default file extension is ***ply.ini***. Refer to **Chapter 5** on *Data Playback* for more information on the playback configuration file.

3.4.1.4 File → Properties

This command opens the Properties window, **Figure 3-4**.

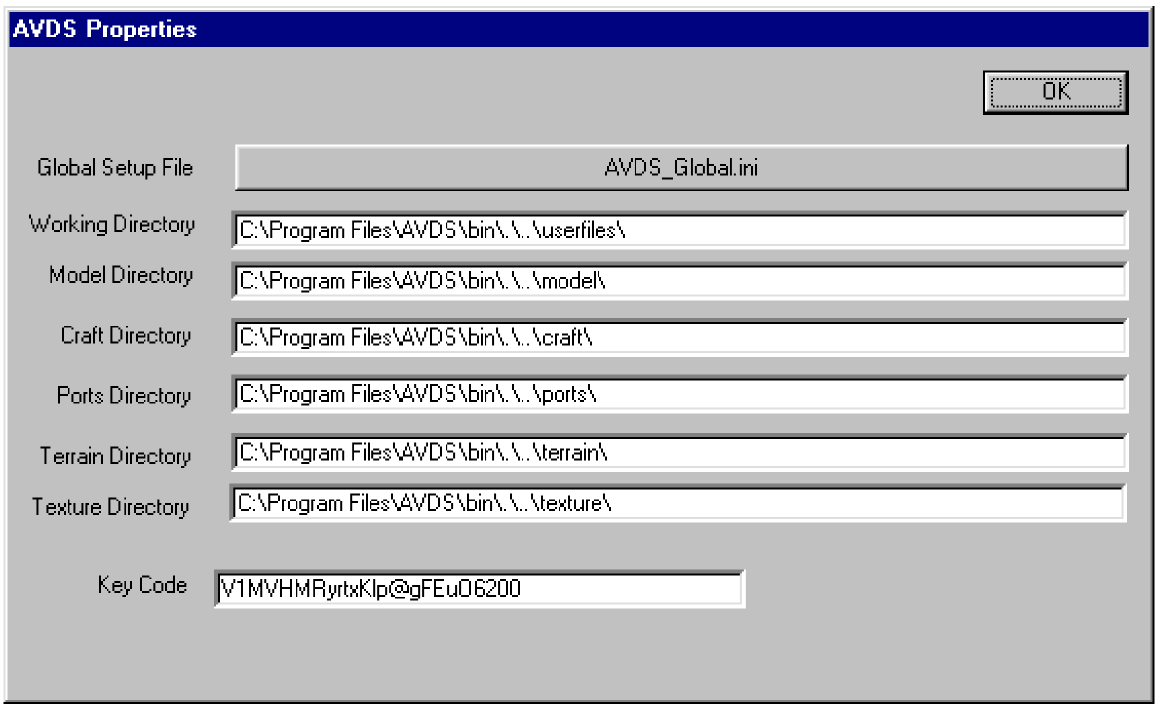


Figure 3-4 AVDS Properties Window

3.4.1.5 Mode Menu

The **Mode** menu lets the user choose the mode of operation for AVDS and provides a START and STOP command. This menu performs the same functions as the **Main toolbar**, **Section 2.2.1**

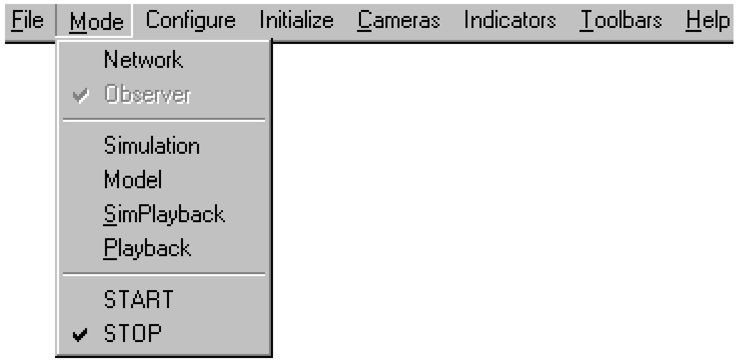


Figure 3-5 Mode Menu

3.4.1.6 Mode → Network

This option selects networking mode and initializes the network. For network configuration information see, Section **3.4.1.26**. When Networking is active vehicle packets are sent to the local network during all of the modes, except Observer mode.

3.4.1.7 Mode → Observer

Observer mode enables the user to display other aircraft that are sending information on the network, including their cockpit viewpoint. Selecting the Observer button causes identification information about other aircraft on the network to be displayed in the aircraft image menu.

3.4.1.8 Mode → Simulation

This mode is for interactive simulation.

3.4.1.9 Mode → Model

Model mode allows the user to manipulate the aircraft image with special commands. This gives the user the ability to visualize aircraft attitude with respect to the various axes systems. Model mode also allows the user to explore the terrain database.

3.4.1.10 Mode → SimPlayback

Simulation & Playback (**SimPlayback**) - This mode is used for simultaneous interactive simulation and playback, i.e., the user can fly the aircraft interactively in the same environment as a pre-recorded aircraft.

3.4.1.11 Mode → Playback

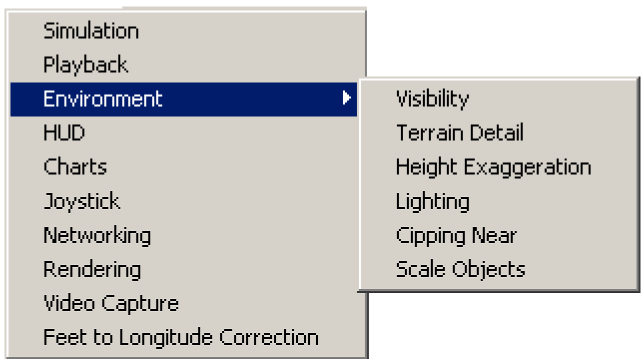
Playback mode is the playback or animation mode.

3.4.1.12 Mode → START

Start Button - Pressing the Start button begins simulation, playback, observer, or model modes. When the configurations are complete and all selections are made, press the Start button to activate the selected mode.

3.4.1.13 Mode → STOP

Stop Button - Pressing the Stop button ends simulation, playback, observer, or model modes.



3.4.1.14 Configure Menu

Figure 3-6 Configure Menu

3.4.1.15 Configure → Simulation

Selecting **Simulation** brings up the Simulation Configuration window. Refer to Chapter 4, *Interactive Simulation*, for details about this window.

3.4.1.16 Configure → Playback

Selecting **Playback** brings up the Playback Configuration window. Refer to **Chapter 5**, *Data Playback*, for details about data playback and this window

3.4.1.17 Configure → Visibility

Displays a window for adjustments to the horizontal visibility, in miles, see **Figure 3-7**. TIP: Reducing the visibility increases the refresh rate of the graphics.

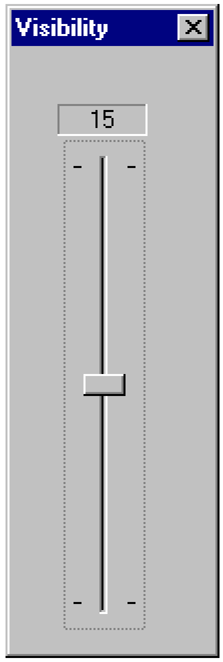


Figure 3-7 Visibility dialog

3.4.1.18 Configure → Terrain Detail

Displays a window for configuration of terrain rendering, see **Figure 3-8**. On this window is a slider to change the terrain detail level. Adjustments to the terrain detail level change the number of polygons that are use to form the terrain. This level is adjustable from 0 to 10. Also on this page is a button to turn terrain rendering off and on, **No Terrain**. The bottom button on the page toggles between smooth (Gouraud) shading, and flat shading. Each Flat shaded polygon is one color. With a smooth shaded polygon, the vertex colors are interpolated across the polygon. Since the speed that the terrain is rendered is a major factor in determining the frame rate of the simulation or animation, the user can greatly increase the frame rate by reducing the terrain detail.

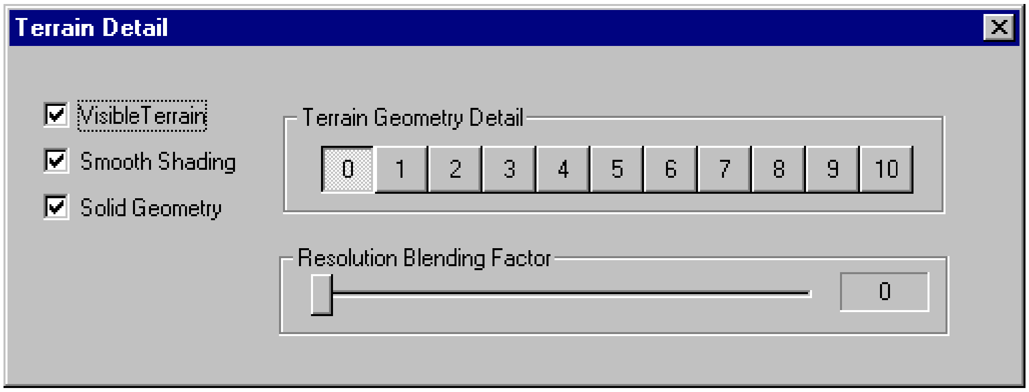


Figure 3-8 Terrain Detail dialog

3.4.1.19 Configure → Height Exaggeration

Displays a window for changing the terrain height exaggeration, see **Figure 3-9**. The range is between 1 and 4.

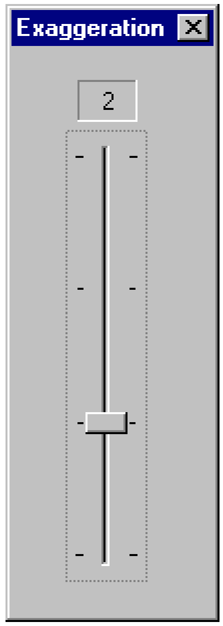


Figure 3-9 Terrain height Exaggeration dialog

3.4.1.20 Configure → Lighting

This menu selection brings up a window to configure lighting properties. The sliders for **Craft Lighting**, move the position of the light source for the aircraft. The **Color** button brings up a dialog window for choosing a new color. This is the color of the aircraft light source. The ambient Intensity slider changes the intensity of the light on the terrain and runways. Changes to **Ambient Intensity** on take effect when the **Start button** is pressed.

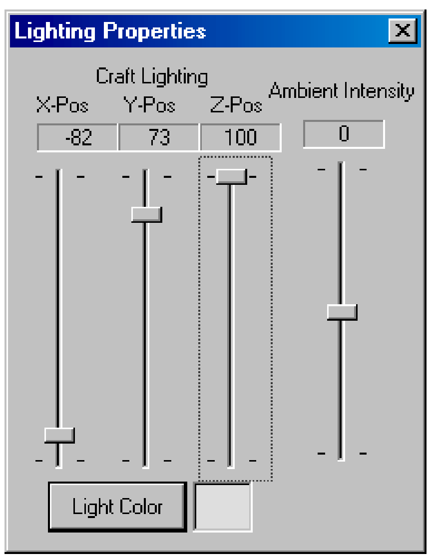


Figure 3-10 Lighting Properties dialog

3.4.1.21 Configure → Clipping Near

Set the position of the near clipping plane with respect to the camera. Object or terrain between the near clipping plane and the camera are not drawn. The distance between the near and far clipping planes determines the distance normalized in the depth buffer. Some graphics accelerator cards use smaller depth buffers than others. Moving the near clipping plane can minimize the effects of a small depth buffer. If the ***Automatic*** box is checked the near plane is automatically set between the object in view and the camera.

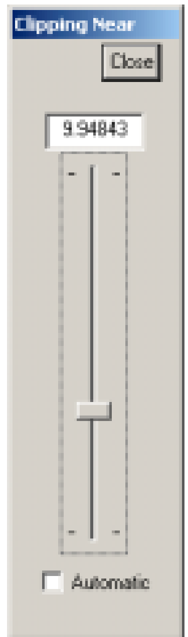


Figure 3-11 Near clipping plane dialog

3.4.1.22 Configure → Object Scale

Scales all of the objects in the scene by the amount selected with the slider control. Note: during Playback mode entities can be excluded from scaling, see section 5.2.2.

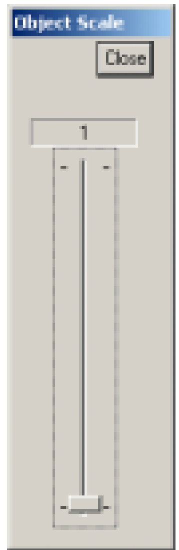


Figure 3-12 Object scale dialog

3.4.1.23 Configure → HUD

This menu selection brings up a window to configure the HUD, see **Figure 3-13**. The HUD"s position and size can be changed using the sliders. These changes are only used when the camera mode is outside the aircraft. The Color button brings up a dialog window for choosing a new HUD color. The **Magnetic Heading Correction** slider allows the user to correct the heading displayed in the HUD for local magnetic variation.

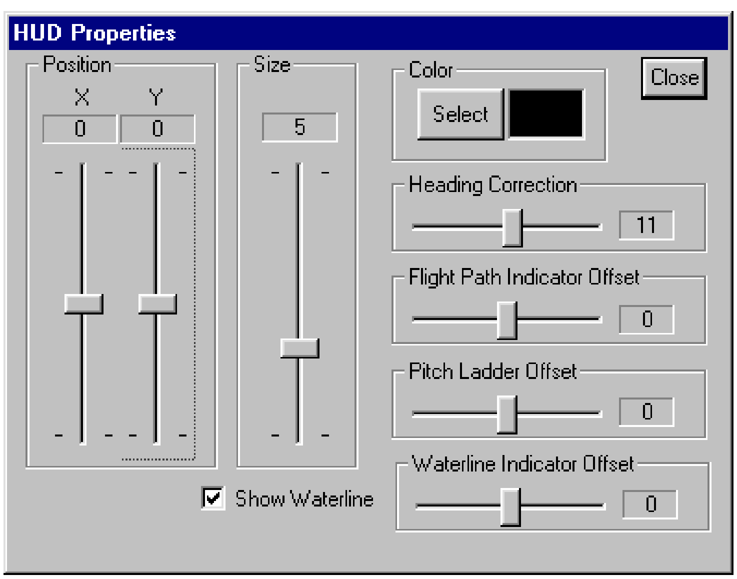


Figure 3-13 HUD Properties dialog

3.4.1.24 Configure → Charts

This selection displays the charts configuration window, see **Figure 3-14** for Simulation settings and **Figure 3-15** for Playback settings.

3.4.1.24.1 Configure → Charts (Simulation)

Pushing the ***Simulation*** button in the **Configuration** Mode area of the Charts Configuration dialog window causes the window to appear as shown in **Figure 3-14.** The various elements in the window have the following meanings:

- **Block** – This is a pull-down menu that contains all of the simulation blocks that are currently active.
- **Available Signals –** These are the signals from the selected block that are available to be plotted in the currently selected chart. Note, once a signal has been added to the Signals to Plot it is removed from the Available Signals list for the selected chart.
- **Signals to Plot –** These are the signals that have been selected for plotting on the currently selected chart.
- **Add** – This button will move the file(s) selected from the Available Signals list to the Signals to Plot list. Note double-clicking a signal in the Available Signals list will also move it to the Signals to Plot list.
- **Remove** – This button will move the file(s) selected from the Signals to Plot list to the Available Signals list. Note double-clicking a signal in the Signals to Plot list will also move it to the Available Signals list.
- **Remove All –** This button moves all of the signals in the Signals to Plot list back to the Available Signals lists.
- **Chart Number –** This is the currently selected chart. It is limited to the value set in Number of Charts.
- **Number of Charts** – This set the number of charts available for plotting signals.
- **Refresh Rate –** This set the refresh rate of the chart in cycles per second (Hz). Note: setting to high of a refresh rate can slow the simulation conversely setting too low of a refresh rate does not allow the display of high frequency phenomena.
- **Window Size** – This is the width of the window in points. Note, new points are added at the rate of set in Refresh Rate. Smaller Window Sizes show less of the time history, but scroll faster. Larger Window Sizes show more time history but scroll slower.

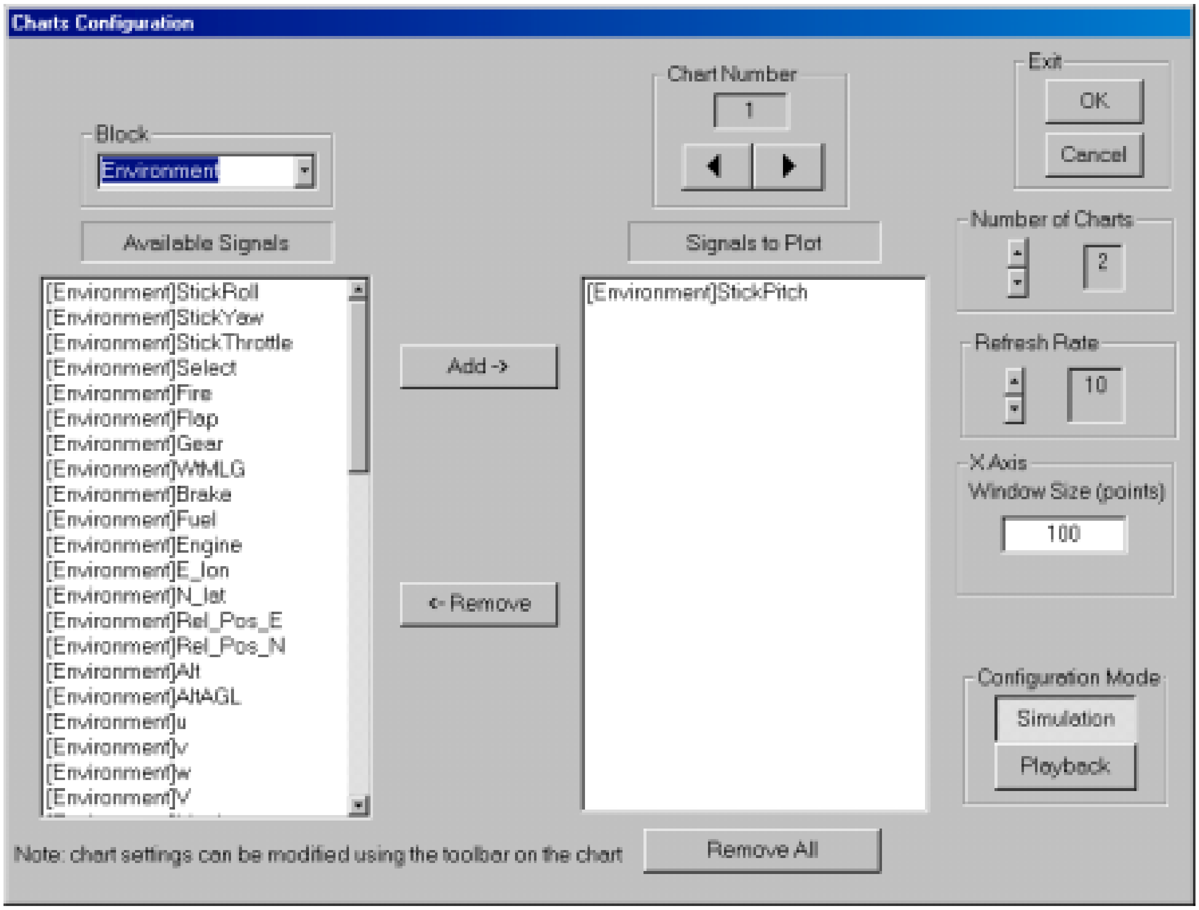


Figure 3-14 Charts Configuration dialog window, while configuring charts for **simulation**.

3.4.1.24.2 Configure → Charts (Playback)

Pushing the ***Playback*** button in the **Configuration** Mode area of the Charts Configuration dialog window causes the window to appear as shown in **Figure 3-15.** All of the elements in the window are the same as those for Simulation except:

- **Entity** – This is a pull-down menu that contains all of the playback entities that are currently loaded.
- **Available Signals** – These are the signals from the selected entity that are available to be plotted in the currently selected chart. Note, once a signal has been added to the Signals to Plot it is removed from the Available Signals list for the selected chart.
- **X separation (msec)** – This is the separation between points plotted in milliseconds. While in Playback mode and not in playback the charts will display the entire time history of the selected data. During playback the width of the chart windows is set to 50 points. In order to change the X resolution of the chart one changes the separation between points. Increasing the separation will increase the time spanned by the chart. Decreasing the time separation will decrease the time spanned by the chart.

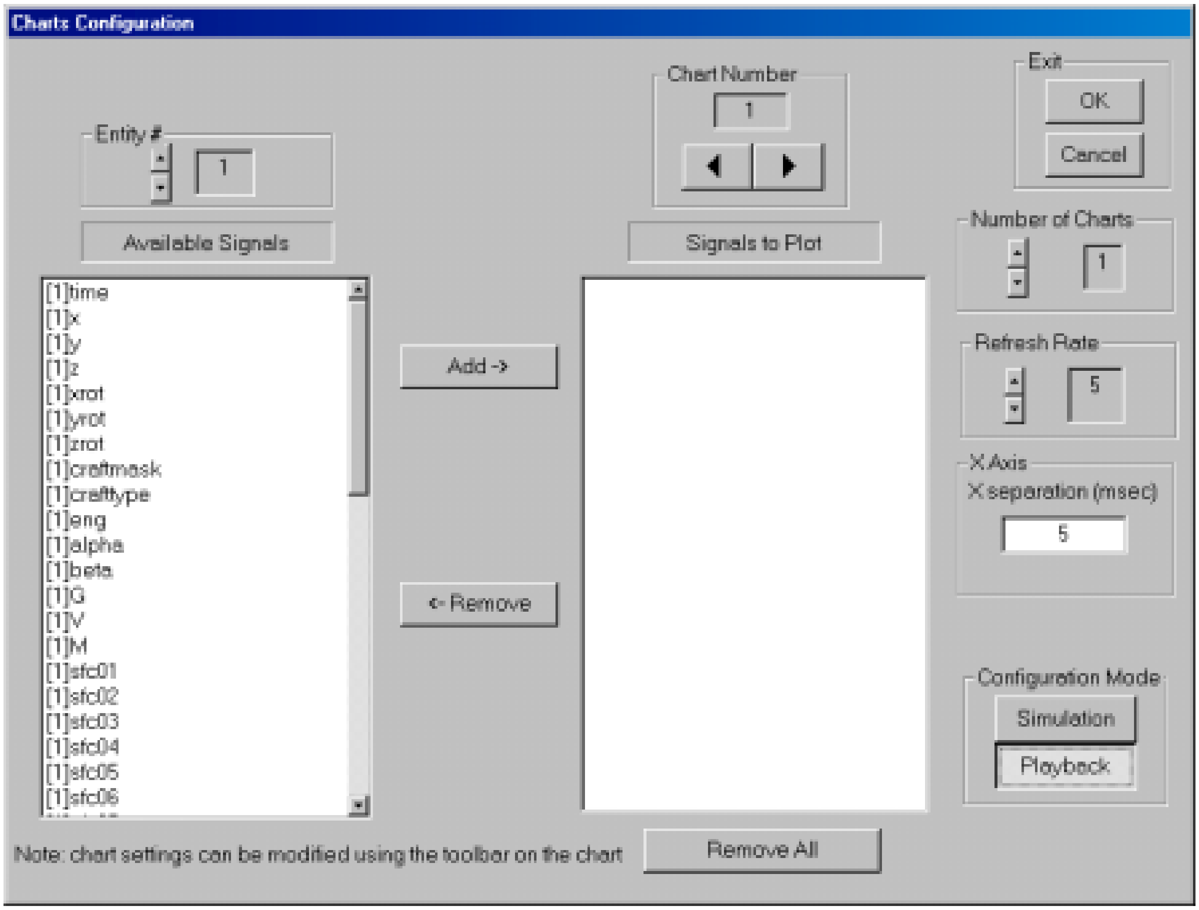


Figure 3-15 Charts Configuration dialog window, while configuring charts for **playback**.

3.4.1.25 Configure → Joystick

This selection displays a window to configure the Joystick, see **Figure 3-16**. AVDS utilizes the MS Windows (NT/95) joystick facilities, so joystick drivers compatible with the windows environment can be loaded and used. AVDS utilizes four analog joystick inputs, nominally X, Y, Z, and R axes; two button inputs, nominally **Select** and **Fire**; and the "**hat**" input selector. The **X-axis** joystick input is mapped to the **Stick Pitch** simulation output, **Y-axis** to **Stick Roll**, **R-axis** to the **Stick Yaw**, and **Z-axis** to the **Stick Throttle** simulation output. Note, for more information on simulation outputs see **Section 4.2**. The **Select** joystick button is mapped to **Select** simulation output and the Fire button is mapped to the **Fire** simulation output. The joystick "**Hat**" input is mapped to a algorithm that "**trims**", or incrementally changes, the **Stick Pitch** and **Stick Roll** simulation outputs. All of the analog inputs are adjusted, based on the state of the sliders in the Joystick Configuration window, by the following algorithm:

Simulation Output = ((Joystick Output Normalized to +/-1.0)+Trim+Bias))*Scale

Table 3-1 outlines the controls in the Joystick window and their effects.

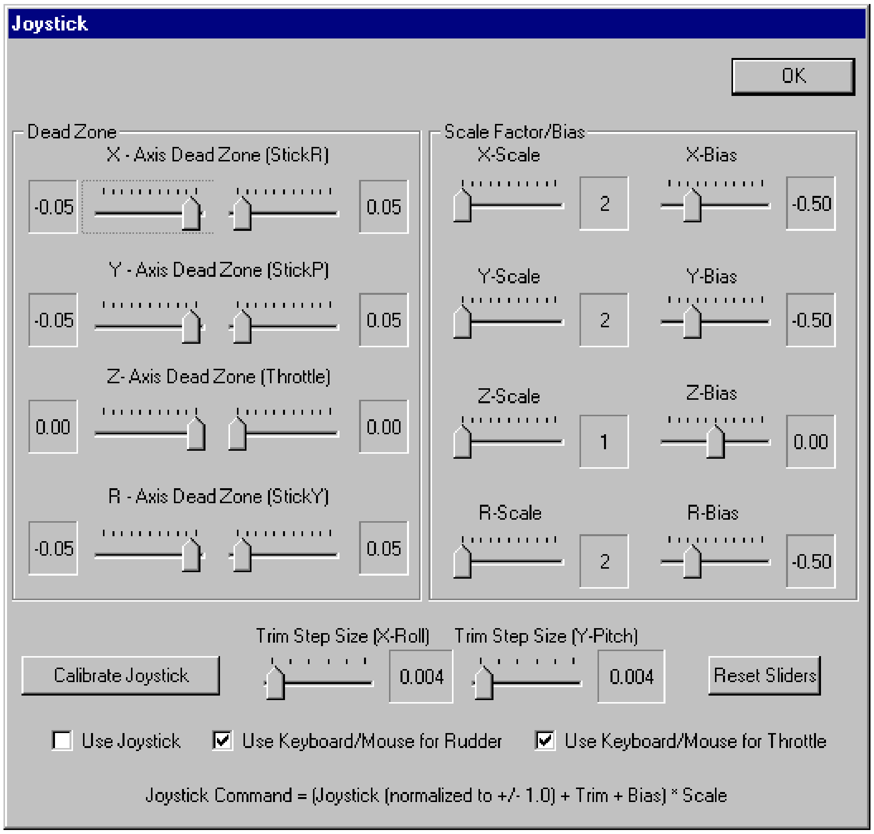


Figure 3-16 Joystick configuration dialog

Table 3-1 Controls in the Joystick Window

Entry	Resulting Action
X, Y, Z, R-Axis Dead Zone	The **dead zone** is the portion of the joystick travel near its center position that doesn"t cause a resulting joystick output. The dead zone can be adjusted for each analog input by moving the appropriate dead zone sliders. The range of these sliders is: LEFT: **-1.00** to **0.00** RIGHT: **0.00** to **1.00**.
X, Y, Z, R-Scale	Adjusts the scale factor for the joystick axis. The range is: **1** to **500**
X, Y, Z, R-Bias	Adjusts the bias factor for the joystick axis. The range is: **-1.0** to **1.0**
Calibrate Joystick	Brings up the MS Windows Joystick Properties dialog window. This window can also be accessed **outside** of AVDS in the **Control Panel** window
Trim Step Size (X-Roll)	Changes the X-axis trim increment step size. This range is: **0.0** to **0.1**
Trim Step Size (Y-Pitch)	Changes the Y-axis trim increment step size. This range is: **0.0** to **0.1**
Reset Sliders	The **Reset Sliders** button returns all sliders to their default positions.
Use Joystick	Checking this block make the joystick active. If this block is unchecked the mouse will be used for joystick functions.
Use Keyboard/Mouse for Rudder	If the joystick is in use and this block is checked, the mouse or keyboard commands are used for **Rudder input**, see Table 2-5
Use Keyboard/Mouse for Throttle	If the joystick is in use and this block is checked, the mouse or keyboard commands are used for **Throttle input**, see Table 2-5

3.4.1.26 Configure → Network

Selecting **Network** brings up the Networking Configuration dialog window, see Figure 3-17. This dialog is used to manage the following:

Simulation ID - change the string used to identify this simulation on the network. The **Simulation ID** is an alphanumeric string that is sent with the networking packets during distributed simulation mode. **Simulation ID** is used to distinguish this simulation from others in the same multicast session.

Installed Multicast Addresses - manages the network-related addresses used for the simulation. **Installed Multicast Addresses** controls which multicast addresses to use to send/receive packets to/from, the port numbers of those addresses, which internet protocol (IP) address packets are sent from, and the port number of those IP addresses. The only required entry is the multicast address and there can be multiple multicast addresses. Essentially, multicast uses IP to transmit packets to many computers at one time. Computers can receive these multicast transmissions if they are set to listen to the multicast address that is being transmitted, and if they are on the same segment of the network that the transmitting computer is on. The format for address entries along with address ranges are listed in the Networking Configuration dialog window, Figure 3-17**.**

Entity Time-Out - set the maximum time that entities remain in the simulation without packet updates.

Entity Caption - provides a method of controlling captions on networked entities. Selecting the *Entity Caption* **Enable** check box causes the display of captions above eace of the networked entities. Pressing the *Entity Caption* **Configure** button brings up ***the Entity Caption Configuration*** dialog window, see Figure 3-18. This dialog window is use to select the caption fonts, and set the size and location of the captions for all of the entities.

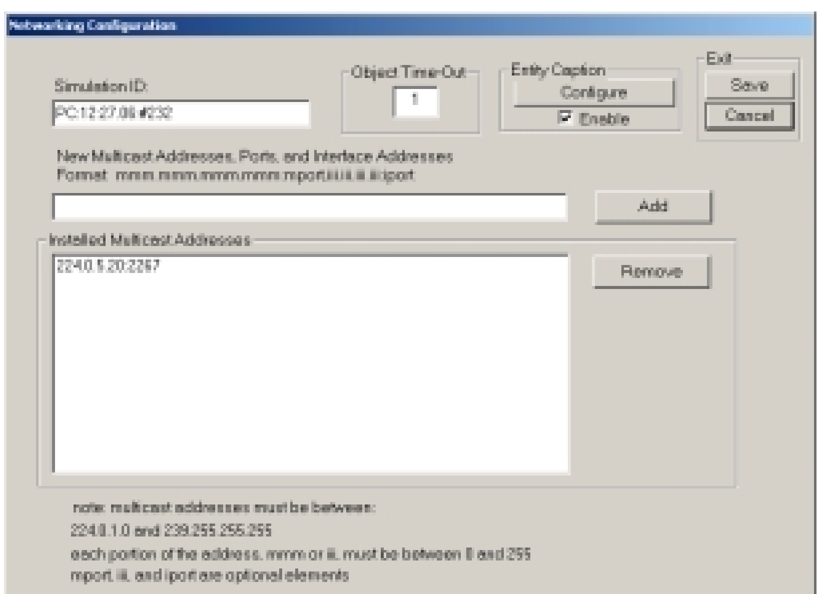


Figure 3-17 Networking Configuration dialog

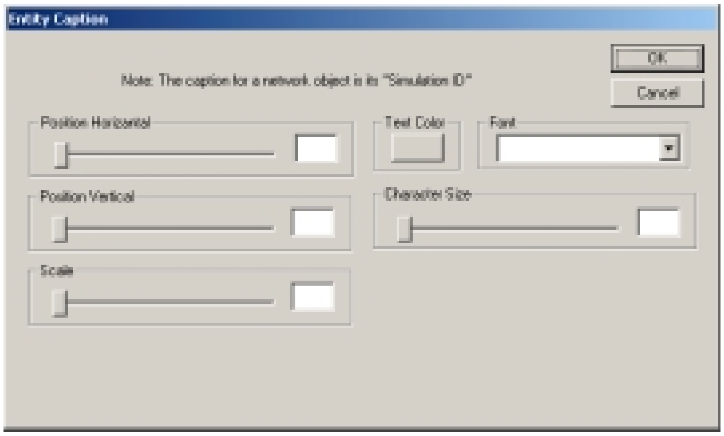


Figure 3-18 Networking Entity Caption dialog

3.4.1.27 Configure → Rendering

Selecting **Rendering** brings up the Rendering Options dialog window, see Figure 3-19. The button labeled "**Smoothing**" toggles vehicle smoothing on and off. The Vehicle smoothing algorithm uses extra normals to the polygons that make up the vehicle graphics to create a uniform transition in lighting between adjacent polygons. This transforms vehicles with relatively few polygons to appear very smooth and finished. The "**Antialiasing**" button controls the application of antialiasing to the scene. Antialiasing removes jagged edges on lines on lower resolution screens. "**Maximum Frame Update Rate**" sets the maximum number of times per second that the scene is redrawn on the screen. This slider should set to the minimum acceptable level for smooth update rate. This frees the CPU to run other processes that are available for execution. "**Dynamics Update Rate**" set the target maximum update rate for the dynamics blocks. If the processor is able it will check for required updates for each of the blocks at this rate.

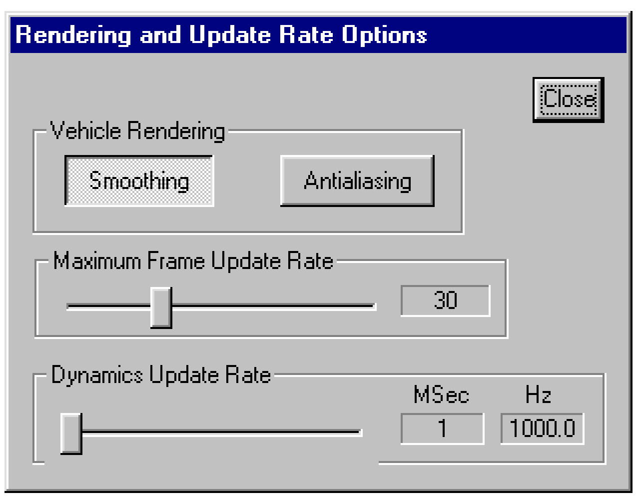


Figure 3-19 Rendering Options dialog window.

3.4.1.28 **Configure → Feet to Longitude Correction**

Selecting this option causes a correction to be applied to all input east/west position data to account for the difference in conversion from feet to longitude, due to changes in latitude. This conversion is necessary when absolute position above the terrain is required. The conversion used is:

$$\textbf{Correction} = 1.0 - |\,(\text{Latitude} + 90.0)\,/\,90.0 - 1.0\,|$$

3.4.2 Initialize Menu

Figure 3-20 Initialize menu

3.4.2.1 **Initialize** → Airport

This is the Airport menu to select the airport from which to fly: This menu contains the airports listed in the ports/**PortsMenu** file.

3.4.2.2 **Initialize → Aircraft Image**

This is the aircraft image menu to select the aircraft image: This menu contains the aircraft images listed in the craft/craftcap file. See the craftcap man page in **Appendix A** for more information on building new aircraft images.

3.4.2.3 **Initialize → Aircraft State**

This menu selection brings up the **Aircraft State Initialization** Window. See **Section 4.5** for more information on aircraft state initialization.

3.4.3 Cameras Menu

This menu allows the user to switch between camera modes and contains the same commands as the Cameras Toolbar, see **Section 2.2.2**

AVDS has three camera views: FlightCamª, ChaseCamª, and TrackCamª. A camera view determines the dynamics of the graphical viewpoint of the user. That is, AVDS allows the user to view the simulation from any angle and any distance, but how the user's viewpoint changes during the simulation is a function of the camera view. In AVDS, the viewpoint can be tied directly to the center of gravity of the aircraft, tied loosely to the center of gravity, or remain fixed with respect to the earth. In all of the camera views, the aircraft is kept at the center of the display.

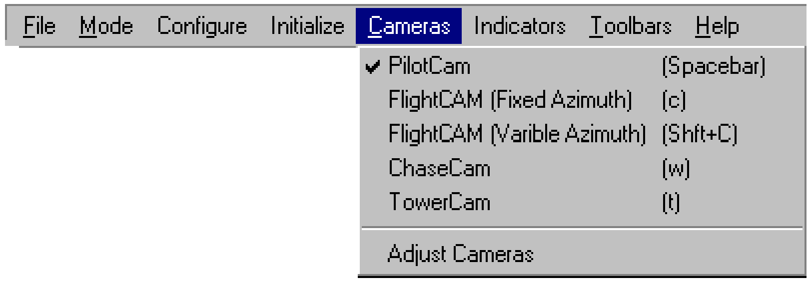


Figure 3-21 Cameras menu

3.4.3.1 Cameras → PilotCam

In PilotCam view, the viewpoint is inside the aircraft. The position of the PilotCam is set in the aircraft"s craftcap file, see the ***pp:*** code in the craftcap an pages in **Appendix A**.

3.4.3.2 Cameras →FlightCam

In FlightCam view, the viewpoint is tied directly to the center of gravity of the aircraft. This gives the user the ability to closely monitor the angular perturbations of the aircraft. As the outside terrain passes by, the aircraft appears as if it were hanging by a string tied to the center of gravity. As shown in the command list in Table 3-2, FlightCam view can also be invoked so that the viewpoint rotates with the heading of the aircraft.

Table 3-2 Commands for FlightCam View

Command	Resulting Action
c	Invokes FlightCam view, tied directly to aircraft center of gravity
C	Invokes FlightCam view and rotates view point with the aircraft heading
... or >	Pan right. Moves position of viewpoint to change the angle between a line from the viewpoint to the center of gravity of the aircraft and the x body axis of the aircraft.
, or <	Pan left. Moves the position of the viewpoint to change the angle between a line from the viewpoint to the center of gravity of the aircraft and the x body axis of the aircraft.
[or {	Move in. Moves the viewpoint, radially, closer to the aircraft center of gravity.
] or }	Move out. Moves the viewpoint, radially, farther from the aircraft center of gravity.
= or +	Pan up. Increases the vertical distance of the viewpoint with respect to the aircraft. Moves the viewpoint parallel to the z body axis of the aircraft.
- or _	Pan down. Decreases the vertical offset of the viewpoint with respect to the aircraft. Moves the viewpoint parallel to the z body axis of the aircraft.

3.4.3.3 Cameras → ChaseCam

In ChaseCam view, the viewpoint is tied loosely to the center of gravity of the aircraft. This view gives the user the ability to closely monitor the linear and angular perturbations of the aircraft. In this mode, the viewpoint appears to be flying in formation with the aircraft. See the command list in Table 3-3.

Table 3-3 Commands for ChaseCam View

Command	Resulting Action
w	Invokes FlightCam view, tied loosely to aircraft center of gravity
[or {	Move in. Moves the viewpoint, closer to the aircraft"s center of gravity.
] or }	Move out. Moves the viewpoint, farther from the aircraft"s center of gravity.

3.4.3.4 Cameras → TrackCam

In TrackCam view, the viewpoint is fixed with respect to the earth. This gives the user the ability to view the flight path of the simulation or playback. TrackCam view provides situational awareness in that the entire area of operations can be viewed at one time.See the command list in Table 3-4.

Table 3-4 Commands for TrackCam View

Command	Resulting Action
t	Invokes TrackCam mode. fixed with respect to the earth. Also moves to the TrackCam set point.
T	Resets the TrackCam set point to the current aircraft position
... or >	Pan right. Moves the position of the viewpoint to change the angle between a line from the viewpoint to the center of gravity of the aircraft and the x body axis of the aircraft.
, or <	Pan left. Moves the position of the viewpoint to change the angle between a line from the viewpoint center of gravity of the aircraft and the x body axis of the aircraft.
[or {	Move in. Moves the viewpoint, radially, closer to the aircraft center of gravity.
] or }	Move out. Moves the viewpoint, radially, farther from the aircraft center of gravity.
= or +	Pan up. Increases the vertical distance of the viewpoint with respect to the aircraft, which moves the viewpoint parallel to the z body axis of the aircraft.
- or _	Pan down. Decreases the vertical offset of the viewpoint with respect to the aircraft, which moves the viewpoint parallel to the z body axis of the aircraft.
z/Z	Zoom in/out. Does not move the camera setpoint location.

3.4.4 Indicators Menu

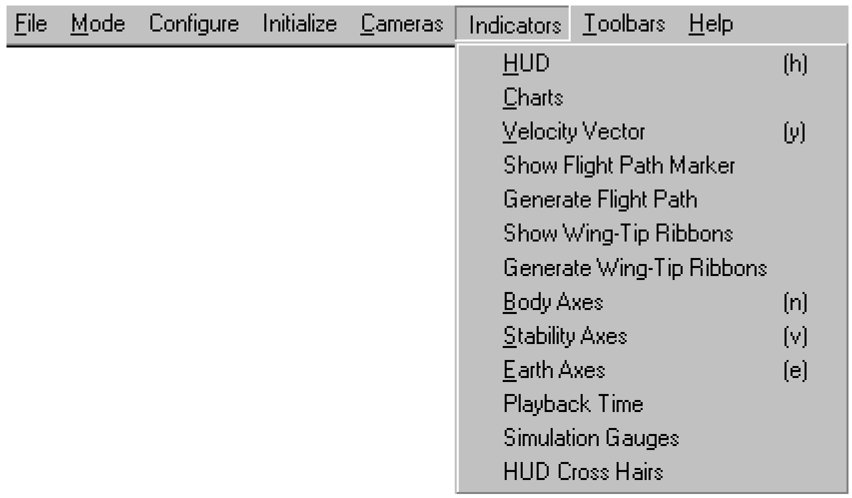


Figure 3-22 Indicators menu

3.4.4.1 **Indicators** → HUD

Turns display of Heads-Up-Display on and off.

3.4.4.2 **Indicators → Charts**

Charts are available for graphical display of simulation outputs. The information presented on the charts can be any of either the simulation outputs or the playback outputs. See the command list in Table 3-5. Right clicking the mouse in the chart area brings up context sensitive menus to set the various parameters of the charts, i.e. axis min/max, colors, etc. See 3.4.1.24 for information on configuring charts.

Figure 3-23 Charts window

Table 3-5 **Command for Strip Charts**

Command	Resulting Action
#	Toggles strip charts on and off.

3.4.4.3 Indicators → Velocity Vector

The velocity vector indicator is available in the interactive simulation and playback modes. The velocity vector is a black pointer emanates from the

aircraft's center of gravity. The length of the vector changes with the total velocity of the aircraft. The angles that the vector makes with the aircraft are based on the angle of attack and the sideslip angle. See the command list in Table 3-6.

Table 3-6 Command for Velocity Vector Indicator

Command	Resulting Action
y	Toggles velocity vector indicator on and off.

3.4.4.4 Indicators → Flight Path Marker

The flight path of the aircraft can be highlighted using a white marker line. Display of this marker is toggled on and off by pressing the letter "i" key. The marker consists of individual line segments that connect the points in space covered by the aircraft. Generation of the flight path marker is toggled on/off by the using the "I" key. See the command list in Table 3-7

Table 3-7 Command for Flight Path Marker

Command	Resulting Action
i	Toggles flight path marker display on and off.
I	Toggles flight path marker generation on and off.

3.4.4.5 Indicators → Wing-Tip Ribbons

These are ribbons that emanate from the aircraft"s wing tips which are defined in the Craft file. The use of wing tip ribbons gives the user a better understanding of the aircraft"s flight path and any maneuvers the aircraft goes through. See the command list in Table 3-8

Table 3-8 Command for Wing Tip Ribbons

Command	Resulting Action
m	Toggles the display of wing tip ribbons on and off.
M	Toggles the generation of wing tip ribbons on and off.

3.4.4.6 Indicators → Body Axes

Body axes are an orthogonal set of axes that are fixed at the center of gravity of the aircraft and are aligned to the aircraft. See the command list in Table 3-9

Table 3-9 Command for Body Axes

Command	Resulting Action
n	Toggles body axes on and off.

3.4.4.7 Indicators → Stability Axes

Stability axes are a set of axes that are fixed at the center of gravity of the aircraft and are aligned to the velocity vector. See the command list in Table 3-10.

Table 3-10 Command for Stability Axes

Command	Resulting Action
x	Toggles stability axes on and off.

3.4.4.8 Indicators → Earth Axes

Earth fixed axes are an orthogonal set of axes that are fixed at the center of gravity of the aircraft and are aligned to the earth. See the command list in Table 3-11.

Table 3-11 Command for Earth Fixed Axes

Command	Resulting Action
e	Toggles Earth fixed axes on and off.

3.4.4.9 Indicators → Simulation Gauges

The Simulation Gauges window contains bar plots showing the state of various simulation variables, see Table 3-12.

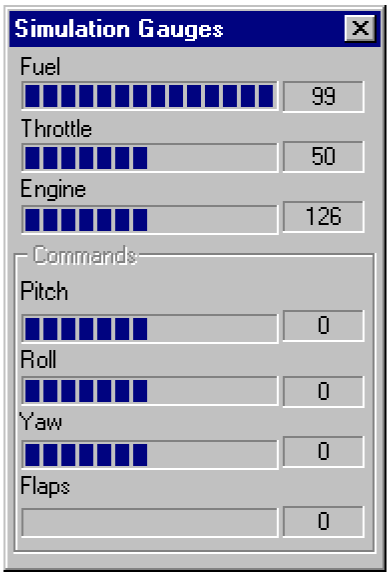


Figure 3-24 Simulation Gauges window

Table 3-12 Simulation Gauge Variables

Label	Variable	Range
Throttle	**% Stick Throttle command**	0 to 100
Engine	**% Engine level**	0 to 100
Fuel	**% Fuel Remaining**	0 to 100
Commands - Pitch	**Stick Pitch command**	-1.0 to 1.0

Commands - Roll	**Stick Roll command**	-1.0 to 1.0
Commands - Yaw	**Stick Yaw command**	-1.0 to 1.0
Commands -Flaps	**Flaps Command (degrees)**	-1.0 to 1.0

3.4.4.10 Toolbars Menu

The Toolbars menu toggle the various toolbars on and off, see **Section 2.2** for more information concerning choices on this menu.

Figure 3-25 Toolbars menu

3.5 Screen Capture

To capture screen graphics, first copy the graphics image using one of the commands in Table 3-13, and then paste the image into a program such as MS Paint

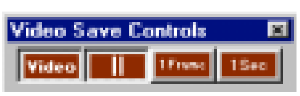
Table 3-13 Command for Earth Fixed Axes

Command	Resulting Action
ALT+PRINT SCREEN	copy an image of the window that is currently active
PRINT SCREEN	copy an image of the entire screen

3.6 Video Capture

AVDS has a video capture utility. AVDS animations are saved to .avi files by capturing frames of the action during Playback mode.

3.6.1 Video Toolbar



 - initialize AVDS for recording video

 - pause/upause video recording

 - record one frame of video

 - record one second of video

3.6.2 Procedures for Saving Videos

1. Switch to  Playback mode
2. Press the  button on the *Video Save Control* toolbar

3. Press the 

button
4. Select the desired playback speed, i.e.  > (1x forward)
5. Unpause the video by pressing the  button.
6. When done recording press the 

button

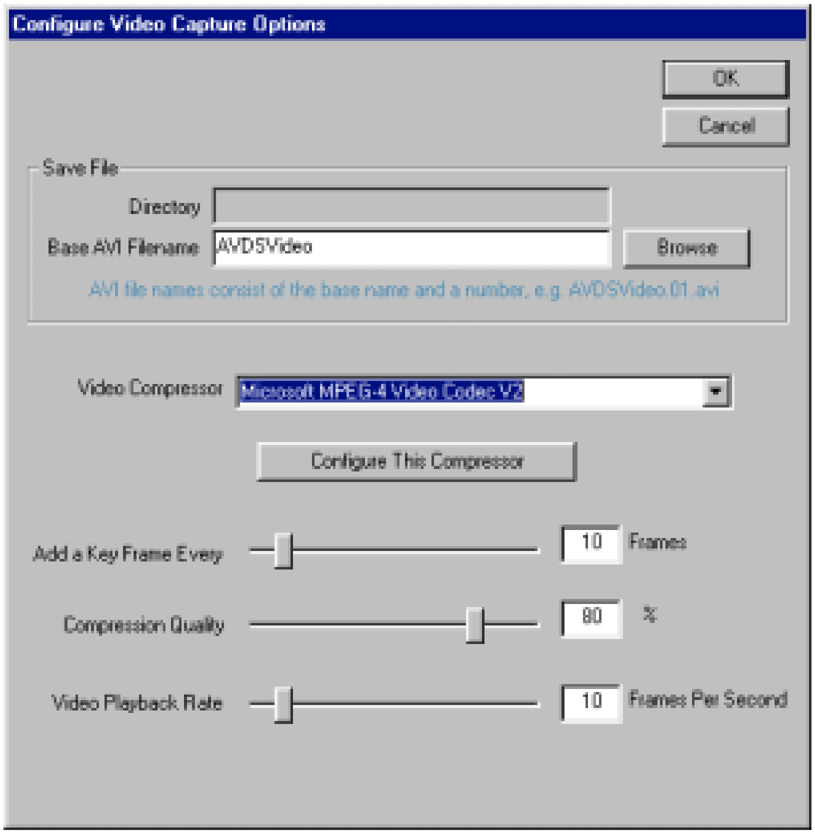


Figure 3-26 Configure Video Capture Dialog Window

3.6.3 Video Configuration Options

Configure->Video Capture

Base AVI Filename: A number is added to this name to create a unique AVI file name. A new file is created for each video.
Video Compressor: Select a CODEC to compress the video.
Add a Key Frame Every: The frequency to save the full image. Only changes in the image are save in between key frames.
Compression Quality: Quality of frame capture.
Video Playback Rate: Number of frames per second to capture.

3.6.4 Tips for Creating Videos

CODEC: Experiment with different CODECs. They make a big difference in both speed of saving videos, and quality/size of saved videos. Many CODECs can be downloaded from support.microsoft.com, search for codec download. One recommended CODEC is "*Microsoft MPEG4 Video Codec V2*".

Frame Dimensions: Choose the smallest frame dimensions possible. The size of the video file is quite a bit smaller when video frame dimensions are small.
Scene Complexity: Scenes with complicated backgrounds contain more information than simpler scenes. When the camera moves, more information needs to be saved. This means that complicated scenes produce larger files than simple scenes.
Camera Dynamics: Keeping the camera stationary keeps the background from changing. This reduces the amount of information that needs to be saved between key frames.
Compression Quality: Lower compression quality reduces the amount of information saved to the video file.
Video Playback Rate: Quite often a video playback rate of 10 frames per second produces acceptable results. Lower playback rates reduce video file sizes. If the frame rate is too high, media players produce jerky playback of the animation.
Key Frame Rate: The more key frames that are saved the larger the video file is. Key frame rate needed depends on the complexity of the scene and the number of moving objects.
Titles: Titles can be added to the video by:

1. pause the video capture.
2. put a document with the title over the AVDS window, i.e. PowerPoint, paint, etc.
3. record a second of video by pressing the 1 second button.
4. remove the title document and resume recording the video.

Combining Video Scenes: Videos with multiple scenes can be constructed by:

1. record the initial scene.
2. pause the video recording
3. change the playback time using the *Playback Time* slider, change cameras, or playback entities.
4. unpause the video recording

Chapter 4 Interactive Simulation

4.1 General

The Interactive Simulation mode gives the user the capability to simulate highly complex aircraft and flight control system models in near real time, while representing the vehicle dynamics graphically and, at the same time, allowing the user to interactively pass commands to the simulated aircraft. The simulation model consists of two parts, model structure and model parameters.

4.1.1 Model Structure

Model structure consists of executable subroutines that define how the model parameters are used to produce the dynamics required to simulate the aircraft. The model structure can either be a user-defined structure or an existing AVDS structure. To incorporate a user-defined structure, the user must compile the user-defined structure with supplied AVDS libraries. This is accomplished with the assistance of a MAKE file and is described later in this chapter. AVDS comes with built-in model structures that allow the user to simulate jet aircraft.

4.1.2 Model Parameters

Model parameters are used to cause the model structure to produce dynamics that mimic those of a given aircraft. The model parameters are specified in a parameter file and read into AVDS at the time the Interactive Simulation begins. This allows the user to change model parameters at any time, between simulation runs, without the need to recompile the program.

The parameters are specified in the parameter file in the form of look-up tables. That is, these parameters can be scheduled based on any of the aircraft dynamic variables, e.g., dynamic pressure, mach, altitude, etc. Furthermore, the look-up table can have up to three dimensions and have associated bias and scale factor parameters.

The Interactive Simulation mode allows the user to use all of the AVDS camera modes, as well as the velocity vector indicator, flight path marker, and strip charts. During the Interactive Simulation, any of the output variables can be recorded and saved to a file. This recorded information can be used for in-depth analysis with other software packages or be animated in AVDS using Playback mode.

4.2 Interactive simulation configuration

Selecting **Simulation** from the Configure menu activates the Aircraft Initialization window, Figure 4-1, see Section 3.4.3. This window allows the user to choose the aircraft and flight control system structures, choose parameter files, make all of the necessary connections between the aircraft flight control system and simulation, specify which outputs are to be recorded, specify which outputs are to be displayed on the strip charts,

choose the recording rate, choose the recording file name, and select ***UserCodeBlocks***. For more information on ***UserCodeBlocks*** see Appendix F.

Note: To make connections between outputs and inputs in this window:

1. Ensure that the desired **input** is **visible.**
2. Select the desired output and drag it over the desired input and drop it.

The entries in the simulation configuration window are listed in Table 4-1

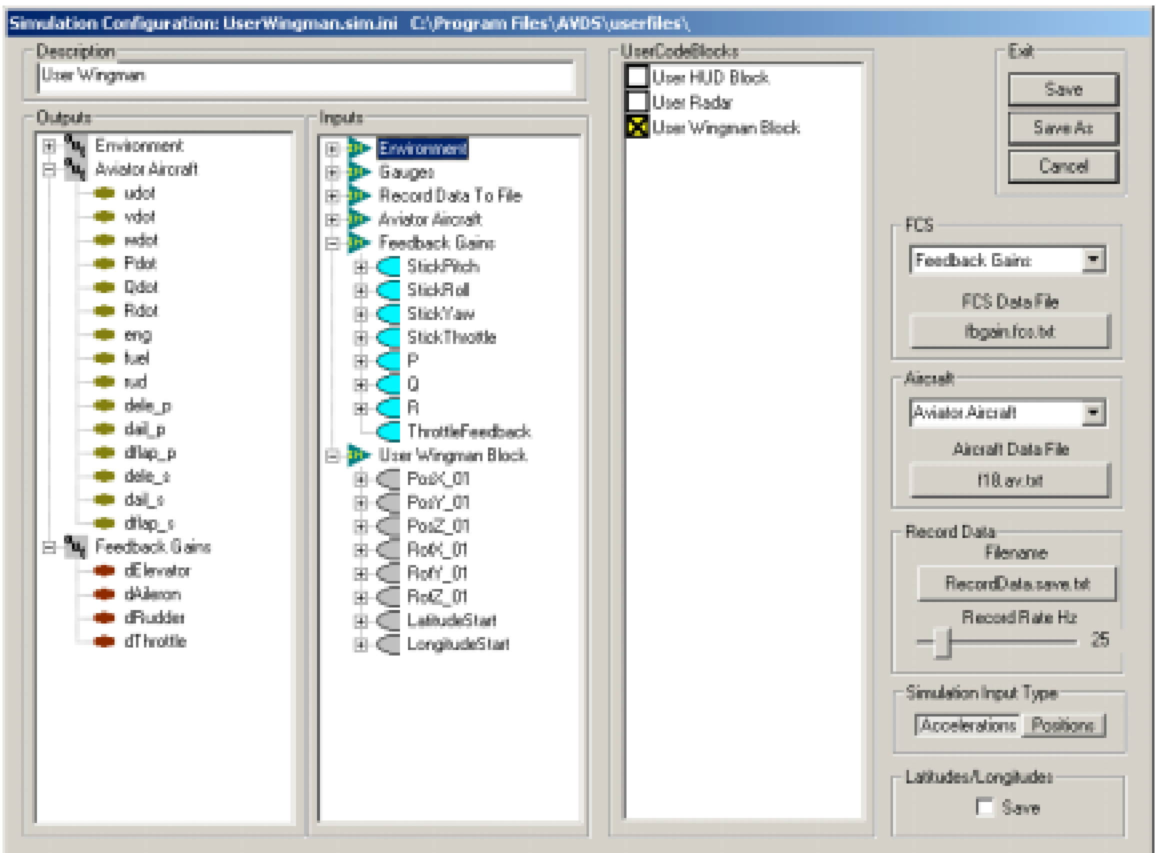


Figure 4-1 Simulation Configuration Window

Table 4-1 Simulation Configuration dialog window entries.

Window Entry	Description
Description	A textual description of the aircraft represented by the simulation configuration window. This description is saved in the Simulation Init File so the user can keep track of aircraft configurations.
Save	Make all of the connections present in the Simulation configuration window active during the interactive simulation, save the settings in the active simulation configuration file and closes the Simulation Configuration window.
Save As	Same as Save, except gives the user the opportunity to choose a new simulation configuration file name.
Cancel	Discards all changes and closes the Simulation Configuration window.
OUTPUTS -	Contains the signals that are available from the AVDS simulation

Environment	environment. See Table 4-2.
OUTPUTS – FCS Block	The output icons represent the signals output from the FCS. The selected type of FCS determines their function and number.
OUTPUTS – AIRCRAFT Block	The output icons represent the signals output from the aircraft model. The selected type of aircraft model determines their function and number.
INPUTS - Environment	These signals are inputs into the AVDS simulation. Note: the items in this list are set based on the Simulation Input Type see Table 4-3 for Accelerations and **Error! Not a valid result for table.** for Positions.
INPUTS - FCS	The input icons below this label represent the inputs to the FCS. The number and types of inputs to the FCS is a function of the selected type of FCS structure.
INPUTS - AIRCRAFT	The input icons below this label represent the inputs to the aircraft model. The number and types of inputs to the aircraft is a function of the selected type of aircraft model structure.
INPUTS - Gauges	These are the inputs to the Simulation Gauges, see Table 4-6.
INPUTS - RECORD	In addition to a default set of signals, these are the signals to be recorded and saved when recording data, see Table 4-7. Note that the signals in Table 4-7 marked default do not appear in the Record list, but are automatically saved during recording.
FCS *(type)*	This menu allows the user to select the structure of the FCS. The structure is either a user defined structure compiled with AVDS, or a predefined structure supplied with AVDS.
FCS Data File	The file that contains the parameter look-up tables for the chosen FCS structure. Pressing this button brings up a file selection dialog window to select the FCS data file.
AIRCRAFT *(type)*	This menu allows the user to select the structure of the aircraft. The structure is either a user defined vehicle structure compiled with AVDS, or a predefined aircraft structure supplied with AVDS.
Aircraft Data File	The file that contains the parameter look-up tables for the chosen aircraft structure. Pressing this button brings up a file selection dialog window to select the Aircraft data file.
Record Filename	The filename of the file for saving the record data. Pressing this button brings up a file chosen dialog window to select the Record file.
Record Rate (Hz)	Frequency to save data in Hz.
Save Lat/Longs	A check in this block causes AVDS to save the latitude and longitude of each position. If the block is un-checked then AVDS saves the starting latitude and longitude and then, for each point, saves the offset from the initial latitude and longitude in feet.

Table 4-2 Environment Output Signals

Signals	Description
StickPitch, StickRoll, StickYaw, StickThrottle	Inputs from the joystick, mouse, or keyboard.
Select	Which weapon is selected, from joystick or keyboard.
Fire	Weapon firing indicator, from joystick or keyboard.
Flap	Indicates if flaps are down or up, from keyboard.
Gear	Indicates if gear is up or down, from keyboard.
WtMLG	Weight on main landing gear indicator.
Brake	Indicates when brakes are being applied, from keyboard.
fuel	Fuel percentage indicator, 0-100%.
engine	Engine power level indicator, 0 - 250
E_lon, N_lat	East Longitude and north latitude of the vehicle in degrees decimal.
Rel_Pos_E, Rel_Pos_N	East and North position of the vehicle, in feet, relative the starting latitude and longitude.
Alt	Absolute altitude of the vehicle in feet.
AltAGL	Altitude of the vehicle above ground level in feet
u, v, w	x, y, z body axis velocities in feet per second.
Velocity	Total velocity in knots.
Mach	Mach number.
Qbar	Dynamic pressure in pounds per square inch.
Rho	Atmospheric density in slugs per cubic foot.
Gs	Acceleration at the pilot station in Gs.
Alpha, Beta	Angle of attack and sideslip angle in radians.
alpha_deg, beta_deg	Angle of attack and sideslip angle in degrees.
P, Q, R	Roll, pitch, and yaw rates in body axis radians per second.
Theta, Phi, Psi	Euler angles in degrees.
dtime	Simulation time step in seconds.
frate	Frame rate. How often the graphics are redrawn, in seconds.
Fgx, Fgy, Fgz	Ground induced accelerations operating through the center of gravity, in feet per second squared (feet/sec^2).
Mgx, Mgy, Mgz	Ground induced angular accelrations about the x, y, and z body axes, in radians per per second squared (rad/sec^2).
craftmask	A hexadecimal parameter that allows the user to change the appearance of the aircraft, see Table 5-1.
craft_type	Zero based index of craft in the craftcap file. That is, this is the number of an aircraft from AVDS's Initialize->Aircraft menu where the numbering starts at zero.
mode	Target lock-on mode during simulation.
Trgt_x, Trgt_y, Trgt_z x, y, and z	Offset to locked on target from center of gravity of aircraft, in feet.
Trgt_az	Azimuth of locked-on target from nose of aircraft in radians.

Trgt_el	Elevation angle of locked-on target from nose of aircraft in radians.
Trgt_vc	Velocity of locked-on target in feet per second.
cam_state	0 – Pilot Camera, 1 – one of the outside cameras
cam_type	0 – Pilot, 1 – ChaseCam, 2 – FlightCam (Fixed Azimuth), 3 – FlightCam (variable Azimuth), 4 - TowerCam
cam_x, cam_y, cam_z	Linear position of the camera in feet.
cam_rx, cam_ry, cam_rz	Angular position of the camera in degrees.

Table 4-3 Environment Input Signals (Entries for Simulation Input Type – ***Acceleration***)

Signals	Description
udot, vdot, wdot	Linear acceleration in the x, y, and z body axes in feet per second squared.
Pdot, Qdot, Rdot	Angular acceleration about the x, y, and z body axes in degrees per second squared

Table 4-4 Environment Input Signals (Entries for Simulation Input Type – ***Positions***)

Signals	Description
xpos, ypos, zpos	Linear positions in the x, y, and z body axes in feet.
xrot, yrot, zrot	Angular rotations about the x, y, and z body axes in degrees.
alpha, beta	Angle of attack and sideslip angle in radians.
G	Acceleration at the pilot station in Gs.
Velocity	Total velocity in knots.
Mach	Mach number.

Table 4-5 Environment Input Signals (Entries for Simulation Input Type – ***Both***)

Signals	Description
InCraftMask	A hexadecimal parameter that allows the user to change the appearance of the aircraft, see Table 5-1.
InCraftType	Zero based index of craft in the craftcap file. That is, this is the number of an aircraft from AVDS's Initialize->Aircraft menu where the numbering starts at zero.
eng	Engine power level indicator, 0 - 250
fuel	Fuel percentage indicator, 0-100%.
sfc01 to sfc15	*These are the inputs to the articulated surfaces defined with the aircraft image in the Craft file, see Appendix A.*
ColorOffsetVehicle	*A signal containing color and transparency definitions that is added to the existing color of all of the polygons of the vehicle image.* *NOTE: this signal contains color information that is constructed in*

	the following manner: *ColorSignal = Transparency*(2^24) + Blue*(2^16) + Green*(2^8) + Red* *Where: Transparency, Blue, Green and Red are values that range between 0 and 255.*
ColorFlightPath	*A signal containing color and transparency data that will be used for the flight path marker.*
ColorRibbonPort	*A signal containing color and transparency data that will be used for the port ribbon marker.*
ColorRibbonStarboard	*A signal containing color and transparency data that will be used for the starboard ribbon marker.*
ScaleX	*A scale factor that will be applied to the X-axis of the vehicle.*
ScaleY	*A scale factor that will be applied to the Y-axis of the vehicle.*
ScaleZ	*A scale factor that will be applied to the Z-axis of the vehicle.*
HUDEntry01	*This signal will appear in the 01 data location of the HUD.*
HUDEntry02	*This signal will appear in the 02 data location of the HUD.*
HUDEntry03	*This signal will appear in the 03 data location of the HUD.*
HUDEntry04	*This signal will appear in the 04 data location of the HUD.*
NOTE: AVDS uses the following definitions: **LINEAR MOTION** **Body axes:** **x – positive forward (through the nose)** **y – positive to the right (through the right wing)** **z – positive down** **Earth fixed axes:** **X – positive north** **Y – positive east** **Z – positive down** **ANGULAR MOTION** ***conventions*:** **pitch - positive nose up** **roll - positive left wing up** **yaw - positive nose right (clockwise)** ***Axes*** **x - Positive rotation / positive pitch** **y - Positive rotation / positive roll** **z - Positive rotation / positive yaw**	

Table 4-6 Environment Gauge Input Signals

Signals	Description
Fuel	0.0 – 1.0
StickThrottle	0.0 – 1.0
Engine	0 - 250
StickPitch	+/- 1.0
StickRoll	+/- 1.0
StickYaw	+/- 1.0
Flap	+/- 1.0

Table 4-7 Environment Record Input Signals

Signals	Description
E_lon, N_lat	**DEFAULT**: East Longitude and north latitude of the vehicle in degrees decimal. (initial position only)
dtime	**DEFAULT**: elapsed time in seconds
Rel_Pos_E, Rel_Pos_N	**DEFAULT**: East and North position of the vehicle, in feet, relative the starting latitude and longitude.
Alt	**DEFAULT**: Altitude of the vehicle in feet.
Theta, Phi, Psi	**DEFAULT**: Euler angles in degrees.
craftmask	**DEFAULT**: A hexadecimal parameter that allows the user to change the appearance of the aircraft, see Table 5-1.
craft_type	**DEFAULT**: Zero based index of craft in the craftcap file. That is, this is the number of an aircraft from AVDS's Initialize->Aircraft menu where the numbering starts at zero.
engine	**DEFAULT**: Engine power level indicator, 0 - 250
alpha_deg, beta_deg	Angle of attack and sideslip angle in degrees.
Gs	Acceleration at the pilot station in Gs.
V	Total velocity in knots.
Mach	Mach number.
sfc01 to sfc15	These are the inputs to the articulated surfaces defined with the aircraft image in the Craft file, see Appendix A.
rec#01 to rec#20	These are "generic" inputs for the user to use to record signals that have not already been accounted for.

4.3 Interactive Controls

During Interactive Simulation, all of the camera display modes are available as well as the velocity vector, charts, flight path marker, and wing tip ribbons. Other than these functions, the control of the vehicle model and control system is accomplished through the use of a combination of joystick, keyboard, and mouse-input devices. When joysticks are not available, the mouse is used in its place.

4.3.1 Joystick

See Table 4-9 and paragraph 3.4.2.9 for more information on joysticks.

4.3.2 Mouse

When a joystick is not connected, the mouse is used for command inputs. See the tables below for Mouse controls with and without joystick. Settings on the Joystick Configuration dialog window cause the mouse to be used to control rudder or throttle when using a joystick, see paragraph 3.4.2.9.

4.3.2.1 Without Joystick

Table 4-8 Mouse Controls without Joystick

Mouse Movement	Resulting Action
Forward/Backward	Commands stick pitch inputs
Left/Right	Commands stick roll inputs
Left Button + Forward/Backward	Changes throttle value
Left and Right Buttons	Changes set point of camera during flight-cam view
Right Button + Left/Right	Command rudder

4.3.2.2 With Joystick

Table 4-9 Mouse Controls with Joystick

Mouse Button	Resulting Action
Left	Move mouse forward/back to set throttle*
Left and Right Buttons	Changes set point of camera during flight-cam view
Right	Move mouse left/right to change rudder**
Forward/Back/Left/Right	Rotates viewing perspective

Caution: *this command is active when the joystick entry **`Use Keyboard/Mouse for Throttle`** is checked.

***This command is active when the joystick entry* **`Use Keyboard/Mouse for rudder`** is checked.

4.3.3 Keyboard

Table 4-10 Keyboard Commands

Command	Resulting Action
Joystick select or s	Select weapon
S	Select no weapon
Joystick fire or Return	Fire selected weapon
Joystick throttle a	Increase throttle
Joystick throttle A	Maximum throttle
Joystick throttle d	Decrease throttle
Joystick throttle D	Minimum Throttle
b	Brakes on/off
f	Flaps down
F	Flaps up
^	Start recording flight data (erases previously saved data, in memory, if it exists
g	Gear up/down

4.4 Recording Data

Data recording can be initiated/re-initiated any time during the Interactive Simulation. Recording is initiated/re-initiated by pressing the ^ (shift-6) key. When the ^ (shift-6) key is pressed during Interactive Simulation, a recording array is initiated in memory, and data is saved to this array. The rate of data saved is obtained by dividing one by the Record Scale factor times the aircraft update period.

Subsequent presses of the ^ (shift-6) key, during the same Interactive Simulation session, will cause the data saved in memory to be lost and new data to be saved to memory. At the end of the Interactive Simulation, when the Esc key is pressed, the data in memory is saved to a file. The filename will be the one displayed on the Record file button in the Simulation Configuration window.

4.5 Aircraft State Initialization

The Aircraft State Initialization window, Figure 4-2, allows the user to select the initial linear or angular positions of the vehicle, as well as the initial linear velocities. Settings in this window are defined in Table 4-11.

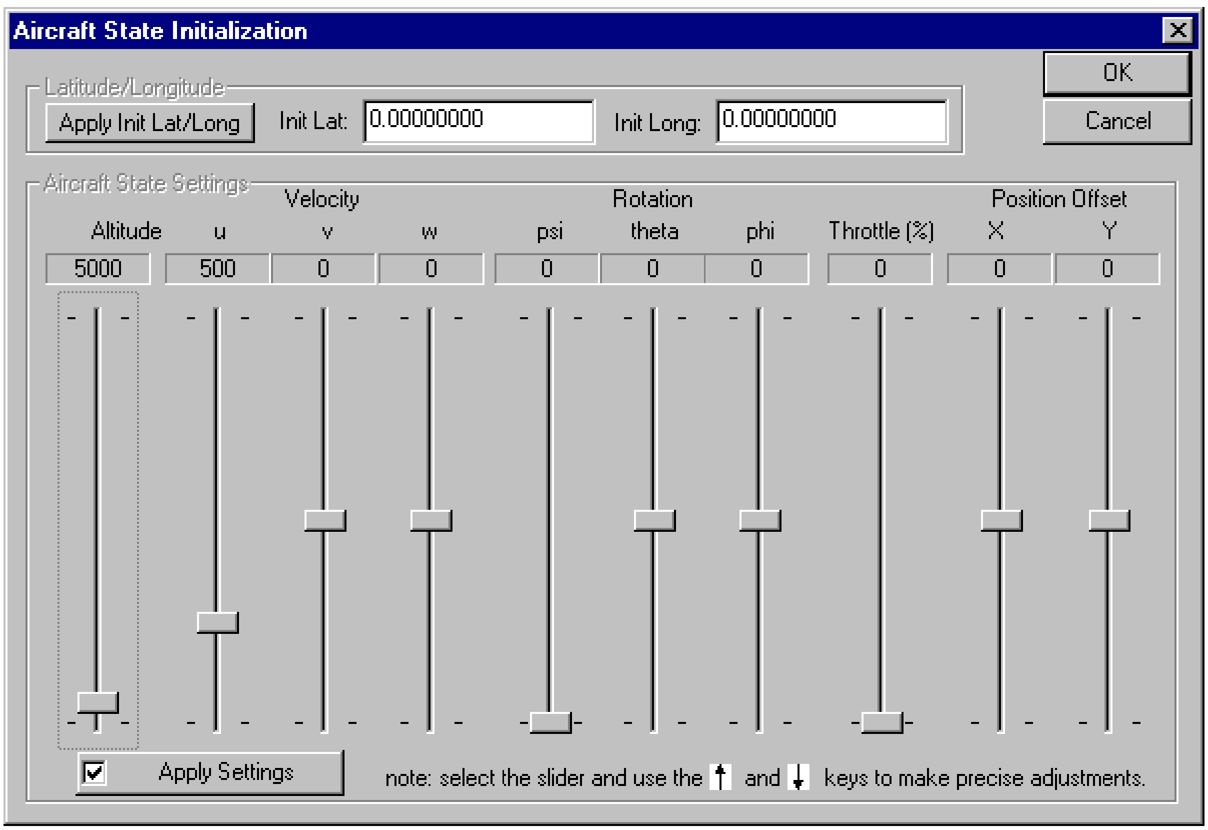


Figure 4-2 Initialize Aircraft State

Table 4-11 Aircraft State Initialization dialog window entries

Control	Variable	Range
Apply Settings	**See below**	up/down
Altitude	**Altitude (feet)**	0 to 100,000s
u	**Forward velocity (knots)**	0 to 2,000
v	**Side velocity (knots)**	-1000 to 1000
w	**Vertical velocity (knots)**	-1000 to 1000r
psi	**Psi Euler angle (degrees)**	0 to 359
theta	**Theta Euler angle (degrees)**	-90 to 90
phi	**Phi Euler angle (degrees)**	-180 to 180
Throttle (%)	**Percent of Throttle position (percent)**	0 to 100
Position Offset X	**East/West Offset from position over the center of the selected airport (feet)**	-100,000 to 100,000
Position Offset Y	**North/South Offset from position over the center of the selected airport (feet)**	-100,000 to 100,000
Apply Init Lat/Long	**see below**[a]	up/down
Init Lat	**Initial Latitude (degrees decimal)**	
Init Long	**Initial Longitude (degrees decimal)**	

If the Apply Settings button is down the settings of the sliders will be used for the initial aircraft state, i.e. if the button is down the aircraft will start in the air over the selected airport, except as noted below, if the button is up the aircraft will start on the ground on the selected airport.

If the Apply Init Lat/Long button is down the simulation will start at the latitude and longitude entered in the Init Lat and Init Long blanks.

4.6 Aircraft Model and Flight Control System Modifications

4.6.1 General

Aircraft models and flight control systems can be modified by either changing look-up table values in the parameter file, or changing the program structure by modifying user defined code blocks.

4.6.2 Look-up Tables

Look-up tables are located in the parameter files. The names of these files are specified in the Interactive Simulation Configuration window. A look-up table is used to model the variations of a given parameter in relation to given variables. For instance, a look-up table can be used to model changes in a parameter such as a stability derivative, due to changes in variables such as angle-of-attack and pitch rate.

4.6.2.1 Format

The format of the Look-Up tables is shown below:

```
# comments
:parameter_name dim1 dim2 dim3 var1 var2 var3 scale_factor bias
var3(1)  var1(1)              var1(2)  …..          var1(dim1)
var2(1)  data(1 1 1)          data(2 1 1)  …..      data(dim1 1 1)
.        .                    .                     ……
var2(dim2)                    data(1 dim2 1)        data(2 dim2 1)  …..
         data(dim1 dim2 1)
……..
var3(dim3)                    var1(1)               var1(2)  …..
         var1(dim1)
var2(1)  data(1 1 dim3)       data(2 1 dim3)  …..   data(dim1 1 dim3)
.        .                    . ……                  .
```

- A "#" must be placed in the first column of any line of comments.
- A ":" must be placed in the first column of any line that contains a table configuration.
- parameter_name - the name of the parameter, specified in the user code block for the particular FCS or aircraft structure.
- dim1,dim2,dim3 - the dimensions of this look-up table
- var1, var2, var3 the independent variables to use as indices in the look-up table. These will appear as inputs to the FCS or aircraft using this parameter file.
- scale_factor - the value obtained from the lookup operation is multiplied by this real number.
- bias - this real number is added to the value obtained from the lookup operation.
- var1_dat_1 ... var1_dat_l, var2_dat_1 ... var2_dat_m, var3_dat_1 ... var3_dat_n - the tabular values of var1, var2, var3 for this parameter. When the actual value of var1, var2, or var3 falls between these tabular values the result of the table look-up operation is obtained through interpolation.
- dat_111 ... dat_lmn - look-up table values.

4.6.2.2 Example

Here is an example using the look-up table in Figure 4-4:

If vel = 500.0, aoa = 45.0 and beta = 90.0, (where vel = velocity, aoa = angle of attack, and beta = side slip angle) then a call to the sample look-up table would return the following: param_A = (-3.0*0.5)+100.0 = 98.5.

```
#This is a Simple Example of a paremeter look-up file
:param_A 4 4 3 vel aoa beta .5 100
-90.0        0.0          100.0        500.0        1000.0
-90.0        0.0          0.3          1.0          3.0
0.0          0.0          0.6          2.0          6.0
45.0         0.0          1.2          3.0          8.0
90.0         0.0          3.0          7.0          10.0

0.0          0.0          100.0        500.0        1000.0
-90.0        0.0          0.3          0.5          1.0
0.0          0.0          1.0          1.0          3.0
45.0         2.0          3.2          2.0          4.0
90.0         4.0          6.0          7.0          5.0

90.0         0.0          100.0        500.0        1000.0
-90.0        0.0          -0.3         -1.0         -3.0
0.0          0.09         -0.6         -2.0         -6.0
45.0         0.0          -1.2         -3.0         -8.0
90.0         0.0          -3.0         -7.0         -10.0
```

Figure 4-4 Simple Look-up Table Example

4.6.3 Adding User Defined Simulation Code Blocks (UDSCB)

To redefine the structure of the FCS or aircraft, the user must add or modify existing UDSCBs. UDSCBs are written in C, or in a language that can interface with C. AVDS is supplied with example UDSCBs that can be modified or used as templates for new models. The header of these models contains the information needed by AVDS to integrate these models into interactive simulations. The UDSCB header has a specific structure and is defined below. Once UDSCBs have been written/modified, they are compiled into object files using the sample MS Visual C++ project files. These object files are dynamically loaded by AVDS at runtime for use in interactive simulations.

When AVDS starts, the **avds.ini** variable 'AVDS_MODEL' points to the directory that contains UDSCBs. This can be the UDSCBs delivered with AVDS in the:

```
AVDS/model
```

directory or a directory selected by the user. This directory must contain a file named '`modelcap`'. The `modelcap` file is used by AVDS to identify UDSCBs in this directory. This file contains a list of UDSCB object filenames.

4.6.3.1 Steps for Adding User Defined Simulation Code Blocks

Refer to **Figure 4.5** for a sample UDSCB.

1. Prepare the UDSCB source code file. Remember to include the file `usr_cblock.h`. For example:

```
#include <usr_cblock.h>
```

2. Define and set the variable '`nBlocks`' to the number of UDSCBs included in this file. For example:

```
int   nBlocks = 1;
```

3. Define the UDSCBs. For example:

```
CBlock FBgainFCS;
```

4. Add these blocks to an execution list. For example:

```
CBlock      *pCBlockx[] = {&FBgainFCS};
```

5. Prototype the new initialization and update functions. For example:

```
void fcs_init();

void fcs_update();
```

6. Declare the UDSCB structure(s); see **Figure 4.5**.
7. Add the simulation code initialization and update functions using the arrays, output[], input[], and param[] for simulation inputs, outputs, and look-up parameters.
8. Compile and link the UDSCB source code file into a dynamic link-library (dll). Note: the sample model files were compiled using Microsoft's Visual C++. The resulting project files are included in the sample directories under the "AVDS/model" directory.
9. Add the name of the resulting dynamic link library file to the modelcap file. For example:

```
my_UDSCB_file1.dll
```

10. Start AVDS, open the Interactive Simulation Configuration window and configure the new model for interactive simulation. See Figure 4-1.

```
#include <usr_cblock.h>
int nBlocks = 1;
CBlock FBgainFCS;
Cblock *pCBlockx[] = {&FBgainFCS};
void fcs_init();
void fcs_update();
/* Add declaration of code blocks */
CBlock FBgainFCS = {
          USR_FCS,                        /* Type             */
          "Feedback Gains",               /* Description      */
          "Ps Rs Ys Ts P Q R",            /* Inputs           */
          "de da dr dt",                  /* Outputs          */
          "k1 k2 k3 cmd1 cmd2 cmd3",      /* Parameters       */
          20,                             /* Update Rate (Hz) */
          fcs_init,                       /* Init Function    */
          fcs_update                      /* Update Function  */
};
enum { STICK_P, STICK_R, STICK_Y, STICK_T, ROLL, PITCH, YAW }; /* Inputs */
enum { D_ELE, D_AIL, D_RUD, D_THR }; /* Outputs */
enum { GAIN_P, GAIN_R, GAIN_Y, GAIN_CP, GAIN_CR, GAIN_CY}; /* Parameters
       */
void
fcs_init( input, output, param )
double *input, *output, *param;
{
  output[D_ELE] = input[STICK_P]*param[GAIN_CP]+input[PITCH]*param[GAIN_P];
  output[D_AIL] = input[STICK_R]*param[GAIN_CR]+input[ROLL]*param[GAIN_R];
  output[D_RUD] = input[STICK_Y]*param[GAIN_CY]+input[YAW]*param[GAIN_Y];
  output[D_THR] = input[STICK_T];
}

void
fcs_update( input, output, param )
double *input, *output, *param;
{
  output[D_ELE] = input[STICK_P]*param[GAIN_CP]+input[PITCH]*param[GAIN_P];
  output[D_AIL] = input[STICK_R]*param[GAIN_CR]+input[ROLL]*param[GAIN_R];
  output[D_RUD] = input[STICK_Y]*param[GAIN_CY]+input[YAW]*param[GAIN_Y];
  output[D_THR] = input[STICK_T];
}
```

Figure 4-5 User Defined Simulation Code Block Header Information

4.6.3.2 User Defined Simulation Code Block Header Information

Refer to Figure 4-6.

- *Blockname* - the name of the user code block. This is entered in the `usr_cblock.c` file to notify AVDS of this new code block.
- *Type* - the value in this location is either USR_FCS or USR_AC. The type of the block determines where it is displayed on the Aircraft Initialization page.
- *Description of code block* - a textual description of the code block. This description is displayed in the type menu for the aircraft of FCS on the Aircraft Initialization (?)page.

- *in1 in2 in3 in4 in5 in6* ... - a list of the inputs to this block. These are displayed under `FCS In` or `AC In` on the Aircraft Initialization page.
- *out1 out2 out3* ... - a list of the outputs from this block. These are displayed under `FCS Out` or `AC Out` on the Aircraft Initialization page.
- *param1 param2 param3 param4* ... - a list of the parameters that will have corresponding look-up tables.
- *r* - the update rate in Hz is based on this integer number. The actual update may vary depending on the computer and complexity of the simulation model.
- *Init_func* - the name of a function that AVDS will call to initialize the code block.
- *update_func* - the name of the function that AVDS calls at the rate specified by `"r"` to update the simulation.

```
CBlock blockname = { type,                /* Type */
      "description of code block" , /*Description */
      "in1 in2 in3 in4 in5 in6...", / Inputs */
      "out1 out2 out3...",          /* Outputs */
      "param1 param2 param3 param4...",/* Params*/
      r, /*Update Rate (Hz)*/
      init_func                   /* Init Function*/
      update_func                 /* Update Function */
```

Figure 4-6 User Defined Simulation Code Block - Sample

```
/* Add declaration of code blocks* /
CBlockYFFBgainFCS={USR_FCS,/* Type */
      "YF-56 Feedback Gains" ,/* Description */
      "Ps Rs Ys P Q R" ,/* Inputs */
      "de da dr ddifftail",/* Outputs */
      "kpitch kroll kyaw", /*Params*/
      60.                /* Update Rate (Hz) */
      YF_init,           /* Init Function */
      YF_update          /* Update Function */
```

Figure 4-7 User Defined Code Block - Header Information Sample

Chapter 5 Data Playback

5.1 General

Playback mode allows the user to animate the time history of one or many vehicle's linear and rotational motions. Time history data are contained in an array in user provided text files. The columns of the array represent variables to be played back; the rows represent time slices of the data. AVDS loads the data file into memory and then plays back (animates) that data using the timing information specified in the data.

The Playback mode is controlled through the use of commands from the playback toolbar or keyboard. During playback, the action can be displayed in real time, faster than real time, slower than real time, frozen, or in single time steps. In addition to animating the motions, additional data columns provided in the user text file can be displayed synchronously on strip charts. While in Playback mode, the user is able to view the action from all of the AVDS camera modes and also display the flight path, axes, ribbons, and velocity vector of the vehicle.

5.2 Playback Configuration Window

5.2.1 General

This window is invoked by selecting “Configure->Plaback” from the AVDS main menu. The Playback Configuration dialog window, shown in Figure 5-1, allows the user to specify the data file, indicate dynamics and extra data columns, and save the playback configuration to a file.

5.2.1.1 Playback Data File

Playback Data File is an ASCII file that contains the data to be animated. The first lines in the file are entries for initial latitude, longitude, and enumeration ID. These entries are optional. The remaining lines in the file are the time slices of data. The data columns can be in any order. As explained below the order of the columns is set-up in AVDS using the Playback Configuration dialog window, see 5.2.2.5.

Initial Latitude/Longitude: At the beginning of the playback file the user can specify entries for the aircraft's initial latitude and longitude in degrees decimal. This is optional and if the initial latitude and longitude are not specified they are set to those of the selected airport. The initial latitude and longitude is the point where the animation starts; in most cases, it is not critical where this point is. An initial latitude and longitude can be obtained by flying an aircraft in Simulation mode to the desired location and then starting the Record function at that point; this will save that latitude and longitude to the save file. If the latitude and longitude are specified as *9999.9* and *9999.9*, AVDS will interpret the x and y data as latitude and longitude. Examples of latitude and longitude entries are:

```
% LAT 37.793625
```

```
% LONG -122.32856
```

and to signal the use of latitude and longitude in the x/y entries:

```
% LAT 9999.9
% LONG 9999.9
```

Enumeration ID: An enumeration ID can be entered to select a craft to be associated with the data. These IDs are optional entries in the craft files, see the entry for ID in 0. An example of the enumeration entry is:

```
% ID 23 45 21 22 45 77 222 33
```

Comments: Except for lines containing LAT, LONG, and ID, line that start with *%*, *#*, or *$* are ignored, i.e. percent, hash, or dollar symbols.

Example Data File:

```
% LAT 37.793625
% LONG -122.32856
% ID 23 45 21 22 45 77 222 33
%time  xpos ypos Alt zrot craftype      ScaleFactor
0      50000  10000  4500   263.6598083  0 10
9.9    32000  8000   4500   263.6598083  0 10
10     32000  8000   4500   240.1010982  0 10
19.9   20000  1100   4500   240.1010982  0 10
```

5.2.2 Playback Configuration Window Entries

5.2.2.1 Entity Data File Name

Pressing this button brings up a file selection dialog window to select a data file for this entity.

5.2.2.2 Entity Data File Path

This is a read-only space for displaying the path of the entities data file.

5.2.2.3 Entity Caption

These are two controls that configure captions that can be displayed with each entity. Selecting the *Entity Caption* **Enable** check box causes the selected caption to be displayed above the entity. The *Entity Caption* button displays the caption’s text. Pressing this button brings up the **Entity Caption** configuration dialog window. This window is used to select the caption text, select the font, and select the size and location of the caption relative to the entity, see Figure 5-2.

5.2.2.4 Entity Configuration

File Column - The icons in this windowpane represent the columns of the data file. To configure AVDS to playback a data file, these icons are dragged to the appropriate icon in the **Dynamics Items** and **User Data Items** windowpanes.

5.2.2.5 Dynamics Items

These are the data items directly used by AVDS for animation, see Table 5-12.

5.2.2.6 User Data Items

Allows the configuration of extra data to be displayed during animation. The user specifies the a five character label, which is displayed with the data

- Change Selected Dynamics Item - To change the sign on a dynamics item, select the item and press the **On Selected Dynamics +/-** button.

5.2.2.7 Exit

- Save - Save the settings in the active Playback configuration file and closes the Playback Configuration window.
- Save As - Same as Save, except gives the user the opportunity to choose a new playback configuration file name.
- Cancel - Discards all changes and closes the Playback Configuration window.

5.2.2.8 Entity Selected

- Number - This number is the index of the entity that is currently being configured. Increasing this number beyond the current number of entities with create a new entity.
- Reset – This button resets the current entity to initial values.

5.2.2.9 Entity Time Format

Delta/Elapsed - Allows the user to specify which method of time representation is present in the data file. Elapsed Time is a column of time information that is continually increasing, e.g., 1.01, 1.02, 1.05, etc. Delta Time is a column of time information where each element represents the change in time between data points, e.g., .01, .01, .03, etc.

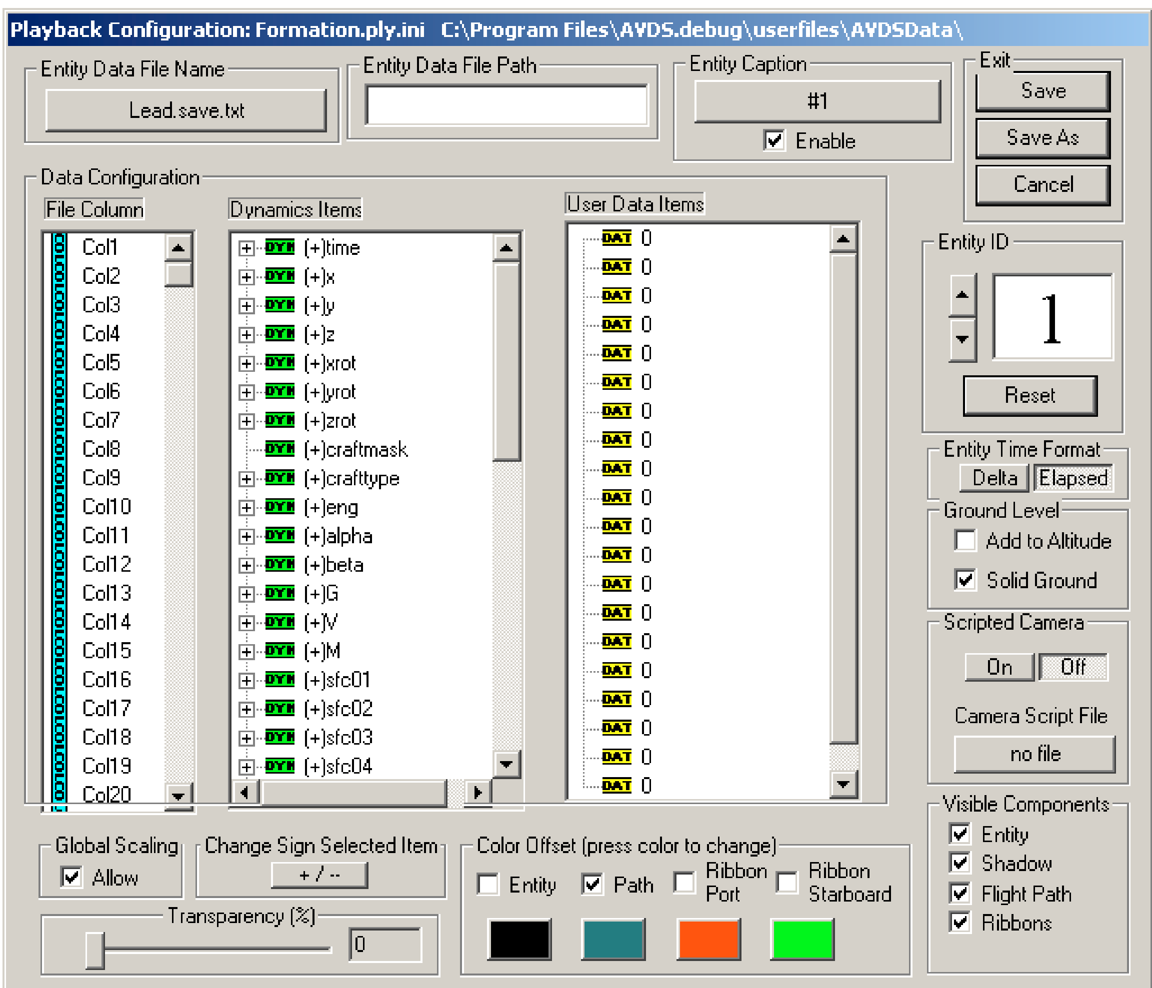


Figure 5-1 Playback Configuration Dialog

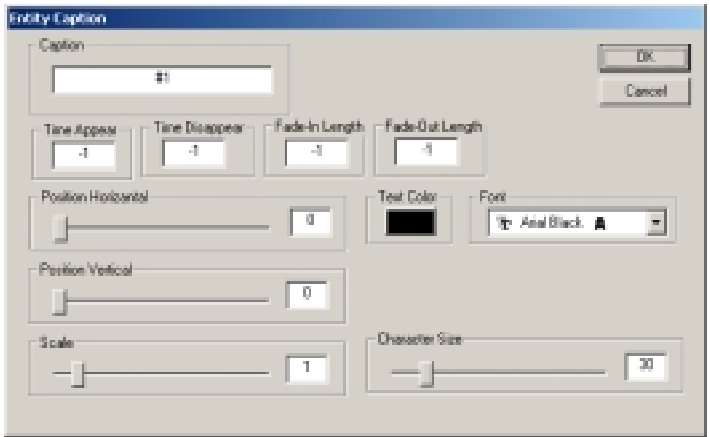


Figure 5-2 Entity Caption Configuration Dialog

5.2.2.10 Entity – Visible

If this block is checked then the entity is visible during playback. If not, the entity will not be present during playback.

5.2.2.11 Entity – Ground Level -> Add To Altitude

If this block is checked then the altitude of the ground level below thw entity is added to the entities altitude, i.e. if the entities altitude is 0.0 feet then checking this block will cause the entity to be displayed at ground level.

5.2.2.12 Entity– Ground Level -> Solid Ground

If this block is checked then the entity can not move below the surface of the terrain.

5.2.2.13 Global Scaling

If the "***Allow***" check box is unchecked the entity will not be scaled when the "Object Scale", see section 3.4.1.22, value is changed. To open the Object Scale dialog window choose "Configure->Environement->Scale Objects" from the AVDS menu.

5.2.2.14 Transparency (%)

This slider control selects the transparency of the entity. 0% is no transparency, i.e. opaque. 100% is completely transparent, i.e. invisible.

5.2.2.15 Color Offset

Checking any of the four boxes, Entity, Path, Ribbon Port, or Ribbon Starboard applies the color values, represented by the color box below the entry, to the existing colors in the craft file and to those in the playback data. To choose a new color, click on the box below the component and choose a new color using the color dialog window

5.2.2.16 Visible Components

Checking or un-checking the visible component boxes will cause those components to be shown or hidden, respectively, during playback.

Table 5-12 Dynamic Items

Entry	Description
x,y,z	The describe the linear position, given as an offset, in feet, from the initial latitude and longitude. The axes used in AVDS are X – positive north , Y – positive east and Z – positive down
xrot, yrot, zrot	angular position, given in degrees. These are the Euler angles. The axes used in AVDS are x - Positive rotation / positive pitch, y - Positive rotation / positive roll and z - Positive rotation / positive yaw
craftmask	This is a hexadecimal parameter that allows the user to change the appearance of the aircraft. For example, the craftmask can be used to raise or lower landing gear. Display of an individual aircraft piece is accomplished by setting the binary digit (bit) in the craftmask corresponding to that piece. The bits in the craftmask parameter correspond to categories of polygons in the craftcap file. Table 5-1 shows a map of the bits and the parts of the aircraft that they control. - aircraft with gear down f7ff ffff - aircraft with gear up f7fe 07ff - invisible aircraft 0000 0000
craft type	This is the craft image from the Aircraft Image menu numbered starting at zero. This number can be manipulated to change craft type during the playback.
eng	a column of data describing the state of the engine: 0 is off, 127 is max without afterburners, 128 is min afterburner, and 255 is max with afterburners. This data is only used to animate the engine state.
alpha/beta	angle of attack of the aircraft in degrees. This data is used in the alpha/beta readout positions of the HUD and also, along forward velocity, to display the velocity vector and stability axes.
G	acceleration at the pilot's station expressed in Gs. This data is used only for the G readout in the HUD.
V	total forward velocity in knots. This data is used in the velocity readout position of the HUD and also, along with alpha and beta, to display the velocity vector.
M	Mach number for use in the HUD
SFC01 to SFC15	Articulated surface inputs in radians
select	Data values are used to select weapons on the vehicle.
launch	Valid entries for the data values are 0 and 1. A 1 will cause the selected weapon to be launched. Once an weapon has been launched it will not be reset until playback has been stopped and restarted.
explode	Valid entries for the data values are 0 and 1. A 1 will cause the entity to explode. Once an entity has exploded it will not be reset until playback has been stopped and restarted.
ColorOffsetVehicle	*A signal containing color and transparency definitions that is added to the existing color of all of the polygons of the vehicle image.* *NOTE: this signal contains color information that is constructed in the following manner:* *ColorSignal = Transparency*(2^24) + Blue*(2^16) + Green*(2^8) + Red* *Where: Transparency, Blue, Green and Red are values that range between 0*

	and 255.
ColorFlightPath	*A signal containing color and transparency data that will be used for the flight path marker.*
ColorRibbonPort	*A signal containing color and transparency data that will be used for the port ribbon marker.*
ColorRibbonStarboard	*A signal containing color and transparency data that will be used for the starboard ribbon marker.*
ScaleX	*A scale factor that will be applied to the X-axis of the vehicle.*
ScaleY	*A scale factor that will be applied to the Y-axis of the vehicle.*
ScaleZ	*A scale factor that will be applied to the Z-axis of the vehicle.*
HUDEntry01	*This signal will appear in the 01 data location of the HUD.*
HUDEntry02	*This signal will appear in the 02 data location of the HUD.*
HUDEntry03	*This signal will appear in the 03 data location of the HUD.*
HUDEntry04	*This signal will appear in the 04 data location of the HUD.*

Note: AVDS uses the following Convention for AXES
LINEAR MOTION
Body axes:
x – positive forward (through the nose)
y – positive to the right (through the right wing)
z – positive down
Earth fixed axes:
X – positive north
Y – positive east
Z – positive down
ANGULAR MOTION
conventions:
pitch - positive nose up
roll - positive left wing up
yaw - positive nose right (clockwise)
Axes
x - Positive rotation / positive pitch
y - Positive rotation / positive roll
z - Positive rotation / positive yaw

Table 5-13 Craftmask Parameter Bit Map

Aircraft Mask Category	Hexadecimal
Nothing	0x00000000
Canopy	0x00000001
Engine	0x00000002
Engine_Port	0x00000004
Engine_Starboard	0x00000008
Engine_Opening	0x00000010
Engine_Opening_Port	0x00000020
Engine_Opening_Starboard	0x00000040
Exhaust	0x00000080
Exhaust_Port	0x00000100
Exhaust_Starboard	0x00000200
Fuselage	0x00000400
Gear_Fore	0x00000800
Gear_Port	0x00001000
Gear_Starboard	0x00002000
Gear_Opening_Fore	0x00004000
Gear_Opening_Port	0x00008000
Gear_Opening_Starboard	0x00010000
Articulated_Surface_01	0x00020000
Articulated_Surface_02	0x00040000
Articulated_Surface_03	0x00080000
Articulated_Surface_04	0x00100000
Articulated_Surface_05	0x00200000
Articulated_Surface_06	0x00400000
Articulated_Surface_07	0x00800000
Articulated_Surface_08	0x01000000
Articulated_Surface_09	0x02000000
Articulated_Surface_10	0x04000000
Articulated_Surface_11	0x08000000
Articulated_Surface_12	0x10000000
Articulated_Surface_13	0x20000000
Articulated_Surface_14	0x40000000
Everything	0xF7FFFFFF

5.2.2.17 Scripted Camera

Scripted Camera mode causes the playback to set and change camera positions based on commands loaded from the 'File'. On/Off – Turns the scripted camera command on and off

- Camera Script File – Pressing this button brings up a file selection dialog window to select the camera script file.

The formats for commands in this file are as follows:

- A '#' placed in the first column causes the line to be ignored.
- A ':' must be placed in the first column of a command line.
- Basic syntax for a command line is as follows:

  ': `DeltaTime CamType f1 f2 f3 EntityNum`'

  where,

`DeltaTime` is the amount of time, in seconds, that the command is active

`CamType` is the numerical type of camera: 0=Cockpit, 1=Wingman Cam, 2=Flight Cam, 3=Flight Cam (Follow), 4=Track Cam

`f1, f2, f3,` vary based on the type of camera selected; see Table 5-2 below.

`EntityNum` is the aircraft(entity) that the camera will be locked to.

Table 5-14 Scripted Camera Selections for CamType and f1, f2, and f3

CamType	f1	f2	f3	Remarks
0	n/a	n/a	n/a	
1	Distance	n/a	n/a	Distance in feet
2	Azimuth	Elevation	Distance	Az/Ele in degrees; Distance in feet
3	Azimuth	Elevation	Distance	Az/Ele in degrees; Distance in feet
4	Latitude	Longitude	Height	lat/long in degrees decimal; Height in feet
4	90.0	n/a	Height	Move set point to current aircraft position

5.3 Playback Controls

During animation of user-defined data, action is controlled with the playback toolbar or with the keyboard and mouse using the following commands:

Caution: *The following keyboard commands use the numeric pad on the keyboard.*

- *6 (left arrow)* - increase playback speed by 1X. Causes the step size of the playback pointer to be increased by 1X. This has the effect of increasing the integer playback speed by 1X. Multiple presses of the 6 key will increase the pointer step size multiple times.
- *4 (right arrow)* - decrease playback speed by 1X. Causes the step size of the playback pointer to be decreased by 1X. This has the effect of decreasing the integer playback speed by 1X. When the pointer decreases below zero the motion of the vehicle is reversed. Multiple presses of the 6 key will decrease the counter multiple times.
- *5* - pause. Causes the action to stop. This key toggles between stop action and the operating speed.

- *9 (PgUp)* - forward normal speed. Causes the playback speed to become 1X, forward normal speed.
- *8 (up arrow)* - increase speed scale factor by 0.1 to a maximum of 1.0. The speed scale factor is multiplied by the current speed. A combination of speed and scale factor less than 1.0 is slow motion.
- *2 (down arrow)* - decrease speed scale factor by 0.1 to a minimum of 0.0. After the scale factor reaches 0.0, it is used to single step the data in the direction of the playback speed.
- *7 (home)* - beginning of playback file. Sets the data pointer and, hence, the animated action to the beginning of the playback file.
- *1 (end)* - end of playback file. Sets the data pointer and, hence, the animated action to the end of the playback file.
- *moving mouse* - changes view in Playback mode. When the mouse is moved left or right, the AVDS display pans left or right. Mouse movements up or down causes the AVDS display to pan up or down.

A. File Formats

(craftcap, portscap)

CRAFTCAP

LOCATION
/usr/local/AVDS/craft/craftcap.txt

DESCRIPTION
Craftcap is a data base describing crafts and weapons used by the flight simulator AVDS. Craft and weapons are described in craftcap by giving a set of capabilities which they have and describing the what they look like.

Each entry in the craftcap file describes a craft or a weapon, and is a line consisting of a number of fields separated by ':' character. The first entry for each craft or weapons gives the two names which are known for the craft or weapon, separated by a '|' character. The first name is the most common abbreviation for the craft or weapon. The second name given should be a long name fully identifying the craft or weapon. The first name should contain no blanks and are always in lower case, for consistency in typing; the second name may well contain blanks for readability. Entries may continue onto multiple lines by giving a '\' as the last character of a line, and empty fields may be included for readability. Comments in this file begin with the '#' character as the first character of the line. Comments can appear in the middle of a craftcap entry. To facilitate handling collections of craft, AVDS provides a file inclusion feature. Any line that looks like include "filename" is replaced by the contents of the file filename. The quotes are mandatory and the include files can not be nested.

Capabilities in craftcap are all introduced by two-character codes, and are of three types:

Boolean capabilities indicate that the craft or weapon has some particular feature. Boolean capabilities are simply written between the ':' characters, and are indicated by the word 'bool' in the type column of the capabilities table below.

Numeric capabilities supply numeric information such as the location of a point on the craft. Numeric capabilities are given by the two-character capability code followed by the '#' character and then one or more numeric values separated by spaces. For example, :pt#-4.20 -0.24 0.24: is the numeric entry representing a point at (-4.20,-0.24,0.24).

String capabilities supply a name of option such as the type of missile used by the craft. String valued capabilities are indicated by the word 'str' in the type column of the capabilities table below. String valued capabilities are given by the two-character capability code followed by an '=' sign and then a string ending at the next following ':'. For example, :cn=heatseeking: indicates a heat-seeking missile.

CAPABILITIES

Name	Type	Description
ac	str	Entry of similar craft - should be first entry
av	num	Angle between views used in sorting polygons (degrees)
ca	str	Category of polygons (fuselage,exhaust,etc.)
cl	bool	Close indicates the previous points define a polygon
cm	num3	Center of mass of craft, numbers (X Y Z) in feet
cn	str	Controlled by (pilot|heatseeking|radarguided|bomb)
co	num3	Color of the polygons that follow (Red Green Blue)
ct	num3	Color of transparent polygons follow (Red Green Blue)
d0	bool	Polygons that follow are used when distance > 0 ft
d1	bool	Polygons that follow are used when distance > 250 ft
d2	bool	Polygons that follow are used when distance > 1000 ft
d3	bool	Polygons that follow are used when distance > 4000 ft
dp	bool	Polygons that follow are used when pilots position
DS	bool	Double sided polygons on/off (1-on)
EC	num3	flight path marker color (Red Green Blue)
ep	num3	Exhaust position behind craft, numbers (X Y Z) in feet
ID	num?	Unique enumeration ID for the craft. This is a list of integers.
pa	str	Parent component type
PC	num3	Port wing ribbon color (Red Green Blue)
pp	num3	Pilot's position in craft, numbers (X Y Z) in feet
pt	num3	3D Point on craft, numbers (X Y Z) in feet
Pw	num3	Port wingtip position, numbers (X Y Z) in feet
s[0-9]	str	Stations 0-9 and the weapons at each
SC	num3	Starboard wing ribbon color (Red Green Blue)
SH	bool	Shadow on/off (1-on)
Sw	num3	Starboard wingtip position, numbers (X Y Z) in feet
ra	num6	The rotation axis for an articulated surface, numbers (X0 Y0 Z0 X1 Y1 Z1) in feet
tr	num	Transparency level (0.0 - transparent, 1.0 - Opaque

wh	num	Weight of the warhead in pounds

the following are used for crash detection only:

ix	num	Moment of inertia (Ixx) about the x axis in slugft2
iy	num	Moment of inertia (Iyy) about the y axis in slugft2
iz	num	Moment of inertia (Izz) about the z axis in slugft2
we	num	Unloaded and nonfueled weight of the craft in pounds

Describing Craft

The polygons must be either triangles or quadrilaterals. Quads must be planar. Surface normals are created by taking the cross product of the first two edges (u,v) of the polygon. The normal is described as"

$$(u /||u||) \times (v /||v||)$$

where u=point2-point1 and v=point3-point2. The polygon must not be degenerate; ||u|| != 0 and ||v|| != 0. Since back facing polygons are rejected, infinitely thin objects must be described by two polygons with the same points but opposite ordering to insure their normals are pointing in opposite directions. Long polygons should be broken into smaller polygons near where they intersect other polygons. If the angle between views is negative, a very forgiving polygon sorting algorithm is used. The craft should be oriented so that the Y axis is pointing in the direction of flight, the X axis is pointing to the right (starboard), and the Z axis is pointing up as viewed from the cockpit.

Categories

Groups of polygons are designated to belong to a specific category. Legal categories are "canopy", "engine", "engine-port", "engine-starboard", "engine-opening", "engine-opening-port", "engine-opening-starboard", "exhaust", "exhaust-port", "exhaust-starboard", "fuselage", "gear-fore", "gear-port", "gear-starboard", "gear-opening-fore", "gear-opening-port", "gear-opening-starboard", "articulated-surface-01", "articulated-surface-02", "articulated-surface-03", "articulated-surface-04", "articulated-surface-05", "articulated-surface-06", "articulated-surface-07", "articulated-surface-08", "articulated-surface-09", "articulated-surface-10", and "body-axes".

Similar Craft

If there are two very similar craft, one can be defined as being just like the other with certain exceptions. The string capability ac can be given with the name of a similar craft. This capability should be first so that the capabilities of the similar craft will be copied into the current entry. The capabilities after this entry will override the ones of the similar craft. Normally the first entry of the craft does not have weapons, and the string capability ac describes the different weapon configurations allowed.

Weapons Configuration

There are ten stations (0-9) for mounting weapons. The location is defined

with the point capability (pt) and then the armament at the station (s[0-9]). If more than one point capability (pt) is defined a weapon is defined for each position. The armament must be previously defined. If the armament is "GUN" immediately followed by a integer number of rounds, a gun will be placed at the point and initialized to the number of rounds.

Articulated Surfaces
Any of the "articulated-surface-[01-10]" categories of polygons can be made to rotate about a given axis, see the 'ra' capability above. The definition of a rotation axis is effective for all of the polygons in the same category. The angle of rotation is defined in either interactive simulation or playback modes.

Wing Tip
Wing tip locations are used for the origination points of ribbons that are used to highlight vehicle attitudes during interactive simulation and playback modes. Likewise, the exhaust position 'ep' is used as the origination point of the fllight path marker.

BUGS
Only 256 characters are allowed for string capabilities and AVDS does not check for overflow of this buffer.

PORTSCAP

LOCATION
/usr/local/AVDS/ports/portscap.txt

DESCRIPTION
Portscap is a data base describing ports used by the flight simulator AVDS. Ports and weapons are described in portscap by giving a set of capabilities which they have and describing the what they look like.

Each entry in the portscap file describes a port, and is a line consisting of a number of fields separated by ':' character. The first entry for each ports or weapons gives the two names which are known for the port, separated by a '|' character. The first name is the most common abbreviation for the port or identification. The second name given should be a long name fully identifying the port. The first name should contain no blanks and are always in lower case, for consistency in typing; the second name may well contain blanks for readability. Entries may continue onto multiple lines by giving a '\' as the last character of a line, and empty fields may be included for readability. Comments in this file begin with the '#' character as the first character of the line. Comments can appear in the middle of a portscap entry. To facilitate handling collections of ports, AVDS provides a file inclusion feature. Any line that looks like include "filename" is replaced by the contents of the file filename. The quotes are mandatory and the include files can not be nested.

Capabilities in portscap are all introduced by two-character codes, and are of three types:

Boolean capabilities indicate that the ports or weapon has some particular feature. Boolean capabilities are simply written between the ':' characters, and are indicated by the word 'bool' in the type column of the capabilities table below.

Numeric capabilities supply numeric information such as the location of a point on the ports. Numeric capabilities are given by the two-character capability code followed by the '#' character and then one or more numeric values separated by spaces. For example, :pt#-4.20 -0.24 0.24: is the numeric entry representing a point at (-4.20,-0.24,0.24).

String capabilities supply a name of option. String valued capabilities are indicated by the word 'str' in the type column of the capabilities table below. String valued capabilities are given by the two-character capability code followed by an '=' sign and then a string ending at the next following ':'.

CAPABILITIES

Name	Type	Description
ca	str	Category of polygons (runway|taxiway|marking)
cl	bool	Close indicates the previous points define a polygon
co	num3	Color of the polygons that follow (Red, Green, Blue)
lo	num2	Location of port (Latitude-degrees, Longitude-degress)
pt	num3	3D Point on ports, numbers (X,Y,Z) in feet rel to :lo:
tr	num1	Transparency of polygons (0.0=solid, 1.0=clear)

Describing Ports

The polygons must be either triangles or quadrilaterals. Quads must be planar. No sorting of polygons is done. Polygons are drawn in order. Categories "runway" and "taxiway" must be first.

BUGS

Only 256 characters are allowed for string capabilities and AVDS does not check for overflow of this buffer.

B. AVDS Network Library Functions

(OpenAVDSNetwork, CloseAVDSNetwork, Init_Aircraft, Put_Aircraft, Get_Aircraft)

OPENAVDSNETWORK

Opens one or more than one network connection for sending and receiving AVDS network packets.

int OpenAVDSNetwork(int iNumAddresses,TCHAR *tcaAddresses[])

RETURN VALUES

`Returns an integer with the value 0 on success and 1 on failure.`

REQUIRED HEADER

```
#include <UserNetwork.lib>
```

PARAMETERS

```
iNumAddresses  number of multicast addresses in the tcaAddresses array,
tcaAddresses  an array of multicast addresses in the form of:
       mmm.mmm.mmm.mmm:mport,iii.iii.iii.iii:iport
where:
```

- the mmm's represent the multicast address for sending/receiving packets
- mport represents the multicast port for sending/ receiving packets
- the iii's represent the host address for sending/ receiving packets
- iport represents the host port for sending/ receiving packets
- multicast addresses must be between: 224.0.1.0 and 239.255.255.255
- each portion of the address, mmm or iii, must be between 0 and 255
- mport, iii, and iport are optional elements

REMARKS

Before AVDS packets can be sent and received the network connection must be opened with the OpenAVDSNetwork function.

CLOSEAVDSNETWORK

Closes all open AVDS network connections.

int CloseAVDSNetwork()

RETURN VALUES

```
Returns an integer with the value 0 on success and 1 on failure.
```

REQUIRED HEADER

```
#include <UserNetwork.lib>
```

REMARKS

```
This functions must be called to close AVDS network connections before
new connections can be opened.
```

INIT_AIRCRAFT

Initializes the data structure for an AVDS aircraft.

int Init_Aircraft(Aircraft *aP, char *name,int id,int craft,float *lng,float *lat,float *alt)

RETURN VALUES

Returns an integer with the value 0 on success and 1 on failure.

REQUIRED HEADER

```
#include <UserNetwork.lib>
```

PARAMETERS

*Aircraft *aP* this is a pointer to the aircraft structure (see below) that is used to transfer aircraft dynamics data to the AVDS network packet.

*char *name* the name identifier of the aircraft that the aircraft structure represents.

int id the numerical identifier of the aircraft that the aircraft structure represents.

int craft a number representing the craft type. The craft type is defined by the order of the craft entries in the craftcap file.

*float *lng* initial longitude in degrees. The position of the aircraft is defined in feet relative to this position.

*float *lat* initial latitude in degrees. The position of the aircraft is defined in feet relative to this position.

*float *alt* initial altitude in feet.

```
typedef struct aircraft_struct {
      int state;  /* aircraft state */
      float elev,ail,rud,throttle;/* commands */
      float u,v,w,p,q,r;
      /* Real values in world coordinates */
      double ref_X, ref_Y;               /* lat, long degrees */
      double pos_X, pos_Y;               /* ft. from ref, ft. alt. */
      float pos_Z;
      float pos_dX, pos_dY, pos_dZ;
      float rot_X, rot_Y, rot_Z;         /* deg */
      float rot_dX, rot_dY, rot_dZ;
      unsigned char engine;                    /* throttle position */
      int hostid;
      Packet packet;
      float dt;   /*aircraft delta time*/
} Aircraft;

typedef struct packet_struct Packet;
struct packet_struct  { /* Description          Units */

      double dUserData[16];   /* for user supplied data */

      short current_cm_z;     /* center mass vertical pseudo-half-
inches          */
      short current_cm_y;     /* center mass fore     pseudo-half-
inches          */
```

```
      short current_cm_x;     /* center mass starbord pseudo-half-
inches          */
      int view_Brot_y;  /* View left/right */
      int view_Brot_x;  /* View up/down */
      float Frot_dZ;    /* vertical  velocity   pseudo-third-
minutes/sec    */
      float Frot_dY;    /* north     velocity   pseudo-third-
minutes/sec    */
      float Frot_dX;    /* east      velocity   pseudo-third-
minutes/sec    */
      int Brot_Z; /* vertical  rotation   pseudo-third-minutes
*/
      int Brot_Y; /* north     rotation   pseudo-third-minutes
*/
      int Brot_X; /* east      rotation   pseudo-third-minutes
*/
      float Fpos_ddZ;   /* altitude  accelerate pseudo-half-
inches/sec/sec  */
      float Fpos_ddY;   /* latitude  accelerate pseudo-half-
inches/sec/sec  */
      float Fpos_ddX;   /* longitude accelerate pseudo-half-
inches/sec/sec  */
      float Fpos_dZ;    /* altitude  velocity   pseudo-half-inches/sec
*/
      float Fpos_dY;    /* latitude  velocity   pseudo-half-inches/sec
*/
      float Fpos_dX;    /* longitude velocity   pseudo-half-inches/sec
*/
      int Ipos_Z; /* altitude  linear     pseudo-half-inches
*/
      int Ipos_Y; /* latitude  linear     pseudo-half-inches
*/
      int Ipos_X; /* longitude linear     pseudo-half-inches
*/
      float current_warhead;/*total warhead     pounds of TNT
*/
      float current_fuel;     /* total fuel           pounds
*/
      int render_mask;  /* craft rendering mask MASK_CANOPY,etc.
*/
      int ground_Ipos_Z;      /* height of ground     pseudo-half-
inches          */
      char ground_slope_Y;/* slope of the ground      unitless
(Z/Y*128)          */
      char ground_slope_X;/* slope of the ground      unitless
(Z/X*128)          */
      u_char ground_type;     /* object is over GROUND_TYPE_WATER,etc.
*/
      u_char engine;    /* engine level
      ENGINE_MIN,ENGINE_MAX,etc.  */
      int craft_type;   /* craftcap type  0->INT_MAX craftcap order
*/
      int collide_damage;/* unique craft id     MASK_WING_PORT,etc.
*/
      int collide_craftid;/* unique craft id    unitless
*/
      int collide_hostid;/* hostid        unitless
```

```
*/
      int lockon_hostid;      /* hostid of target     unitless
(0=nolock)         */
      int craft_checksum;/* crafttcap checksum  unitless
*/
      int craftid;      /* unique craft id      unitless
*/
      int hostid; /* hostid         unitless                    */
      short status;     /* status         STATUS_LOCAL,etc.
*/
      char home_port[SIZE_HOME_PORT];     /* Airport name
      string                      */
      char handle[PACKET_HANDLE_SIZE];     /*   string
*/
      int revision;

};
```

REMARKS

This function initializes the aircraft structure. Once the aircraft structure has been initialized, various elements of the structure are set before sending the information using the PutAircraft() function.

PUT_AIRCRAFT

Send the information in an aircraft structure on the local network.

int Put_Aircraft(Aircraft* aP)

RETURN VALUES

Returns an integer with the value 0 on success and 1 on failure.

REQUIRED HEADER

#include <UserNetwork.lib>

PARAMETERS

Aircraft aP* this is a pointer to the aircraft structure (see above) that is used to transfer aircraft dynamics data to the AVDS network packet.

REMARKS

This function processes the information in the aircraft structure and inserts it in an AVDS packet and sends it on the local network.
NOTE: The following networking functionality is invoked through the use of the **dUserData** array in the **packet**.
If the STATUS_NO_EXTRAPOLATE bit is set in the status variable of the packet, as in the following example, then AVDS will not extrapolate the positions and rotations between packet updates. Also the following variables are set using elements from the **dUserData** array in the packet:

- **Velocity** in feet/second => dUserData[USER_DATA_V_FeetSec]
- **Gs** => dUserData[USER_DATA_Gs]
- **Angle of attack** in radians => dUserData[USER_DATA_Alpha_Rad]
- **Sideslip Angle** in radians => dUserData[USER_DATA_Beta_Rad]

Dynamics updates are filtered using the following formula:

$Data_{FIL} = (1.0 - FilterValue)*Data_{NEW} + FilterValue * Data_{LAST}$

Filter parameters range from 0.0 to 1.0

- **Filter Parameter for X** position filters => dUserData[USER_DATA_Filter_X]
- **Filter Parameter for Y** position filters => dUserData[USER_DATA_Filter_Y]
- **Filter Parameter for Z** position filters => dUserData[USER_DATA_Filter_Z]

```
  FOR EXAMPLE:
       f18.packet.status |= STATUS_NO_EXTRAPOLATE;
       f18.packet.dUserData[USER_DATA_V_FeetSec] = 300;
       f18.packet.dUserData[USER_DATA_Gs] = 1.2;
       f18.packet.dUserData[USER_DATA_Alpha_Rad] = 0.087266462599716478846184538424430 6;
       f18.packet.dUserData[USER_DATA_Beta_Rad] = 0.0872664625997164788461845384244306;
#define FILTER_PARAMETER (0.95)
       f18.packet.dUserData[USER_DATA_Filter_X] = FILTER_PARAMETER;
       f18.packet.dUserData[USER_DATA_Filter_Y] = FILTER_PARAMETER;
       f18.packet.dUserData[USER_DATA_Filter_Z] = FILTER_PARAMETER;
```

GET_AIRCRAFT

Receive AVDS aircraft packets from the network.

int Get_Aircraft(void CALLBACK pAircraftCallback (Aircraft *aP),Aircraft * aP)

RETURN VALUES

Returns an integer with the value 0 on success and 1 on failure.

REQUIRED HEADER

#include <UserNetwork.lib>

PARAMETERS

*void CALLBACK pAircraftCallback (Aircraft *aP)* this is a user supplied callback function that is invoked to process the aircraft packets received after the call to Get_Aircraft.

*Aircraft * aP* this is a pointer to the aircraft structure (see above) that is used to transfer aircraft dynamics data to the AVDS network packet.

REMARKS

After a call to Get_Aircraft the user supplied function *pAircraftCallback* is called by the networking library to process the queue of packets that have been received since the last call to Get_Aircraft.

C. Airports

Introduction

This help screen describes the airports available in AVDS. Version 1.0 contains 59 airports; 22 in the San Francisco bay area and 37 in the Hawaiian islands. The AVDS startup screen does not display all of the airports, only the major ones.

Airport Runways

The airport runways in AVDS Version 1.0 are accurately rendered onto the terrain. Complete descriptions of each of the runways may be found in the AVDS ports directory. The information in the ports files includes the location, elevation, owner, runway length and width, and general notes on the runways. Much of this information has been duplicated in the airport descriptions in this help file.
All runways are identified by a number, like those shown in Figure 1. The runway number identifies the compass heading used to land on or takeoff from the runway, in tens of degrees. When two parallel runways, they are identified as either left (L) or right (R). For example, runway 28L in Figure 1 is on a heading of 280 degrees, and is the left runway.
Runway 10R is the complement to runway 28L. That is, runway 10R is approached from the opposite direction; a heading of 100 degrees. Note that the runway headings are identified by magnetic north, which, for San Francisco International Airport, is 16 degrees from true north. Areas alongside the runways, shown in gray in Figure 1, are taxiways and access ramps.

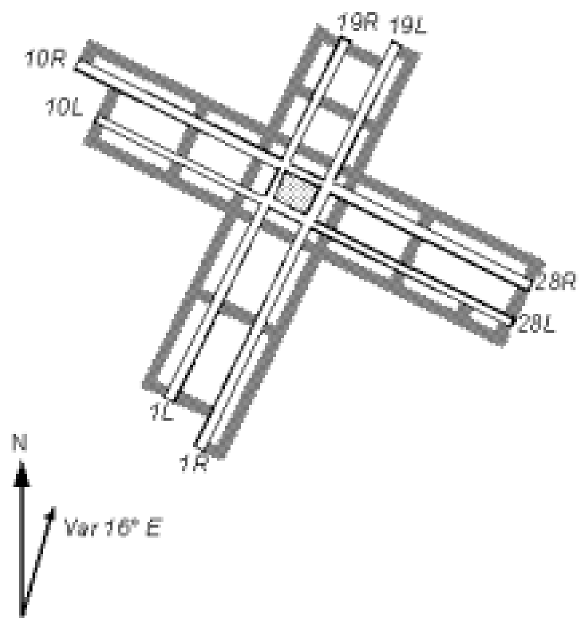


Figure 1 *San Francisco International Airport (SFO).*

Runway Markings

Figure 2 shows a typical airport runway. The runway markings identify the approach direction and the runway name. The runway name, such as 31R in the figure identifies the runway approach heading (310 degrees in this example) and, if there are two parallel runways, whether the runway is the right (R) or left (L).
The preferred touchdown point is between the first and third set of parallel markings.

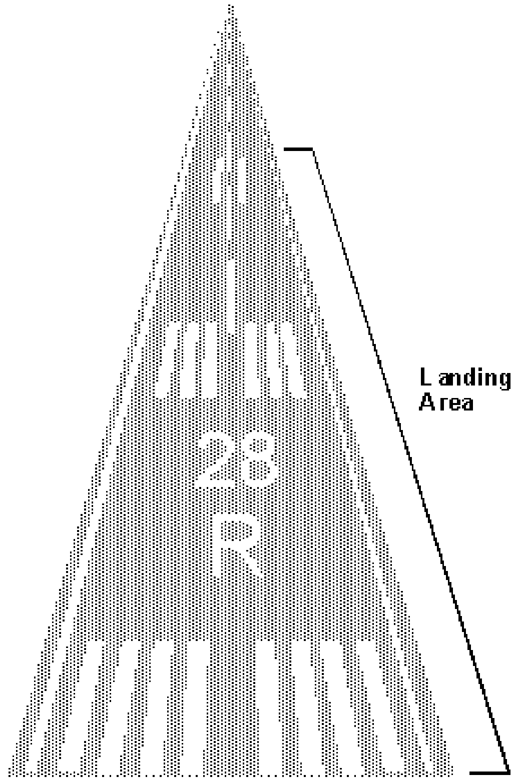


Figure 2 Typical Airport Runway

Airport Maps

San Francisco Bay Area Airports

Figure 3 shows the airports in the San Francisco bay area. For more information, see the following publications:

- VFR Terminal Area Chart, San Francisco
- Airport/Facility Directory, Southwest U.S.

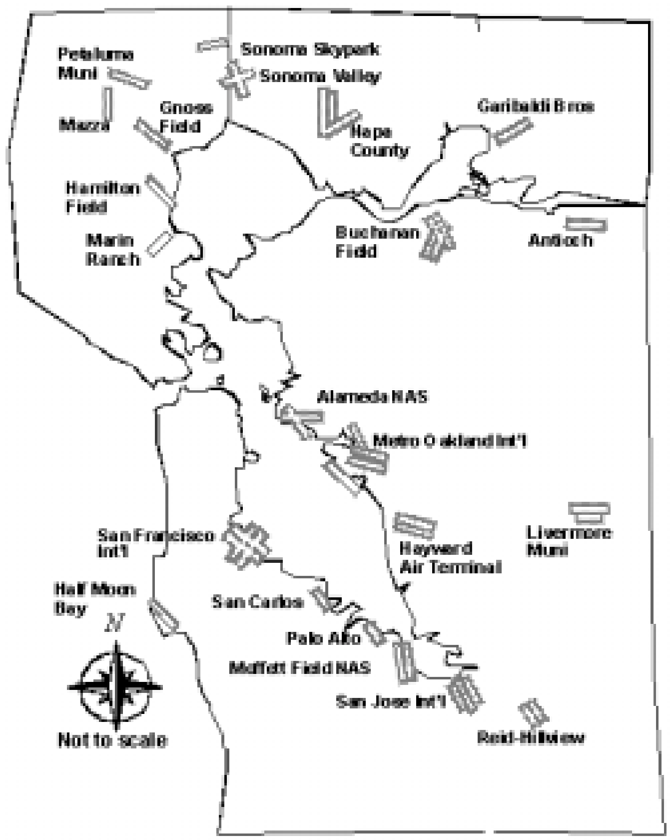


Figure 3 San Francisco Bay Area Airports

Alameda NAS (Nimitz Field)

ID: NGZ	City: Alameda, CA
Elevation: 14'	Owner: US Navy
Rwy: 07 - 25: 7200' × 200'	Rwy: 137 - 31: 8000' × 200'

Antioch

ID: O12	City: Antioch, CA
Elevation: 185'	Owner: Antioch AvÕn Assoc.
Rwy: 09 - 27: 2480' × 33'	

Buchanan Field

ID: CCR	City: Concord, CA
Elevation: 23'	Owner: Contra Costa County
Rwy: 01L - 19R: 4400' × 150'	Rwy: 01R - 19L: 2768' × 75'
Rwy: 14L - 32R: 3951' × 150'	Rwy: 14R - 32L: 2800' × 75'

Garibaldi Brothers

ID: 6Q2	City: Cordelia, CA
Elevation: 20'	Owner: Louis Garibaldi
Rwy: 06 - 24: 2000' × 50'	

Gnoss Field

ID: O56	City: Novato, CA

Elevation: 14' Owner: Marin Countyy
Rwy: 13 - 31: 3300' × 60'

Half Moon Bay

ID: HAF City: Half Moon Bay, CA
Elevation: 67' Owner: San Mateo County
Rwy: 12 - 30: 5000' × 150'

Hamilton Field

ID: SRF City: San Rafael, CA
Elevation: 3' Owner:US Army ATCA ASO
Rwy: 12 - 30: 8000' × 400'

Hayward Air Terminal

ID: HWD City: Hayward, CA
Elevation: 47' Owner: City of Hayward
Rwy: 10R - 28L: 5024' × 150' Rwy: 10L - 28R 3107' × 75'

Livermore Muni

ID: LVK City: Livermore, CA
Elevation: 397' Owner: City of Livermore
Rwy: 07L - 25R: 4005' × 100' Rwy: 07R - 25L: 27009' × 75'

Marin Ranch

ID: CA35 City: San Rafael, CA
Elevation: 5' Owner: Marin Ranch Airport
Rwy: 04 - 22: 2140' × 30'

Mazza

ID: 58Q City: Petaluma, CA
Elevation: 20' Owner: Joseph Tognalda
Rwy: 18 - 36: 1100' × 10'

Metro Oakland IntÕl)

ID: OAK City: Oakland, CA
Elevation: 64' Owner: Port of Oakland
Rwy: 11 - 29: 10,000' × 150' Rwy: 09R - 27L: 6212' × 150'
Rwy: 09L - 27R: 5453' × 150' Rwy: 15 - 33: 3366' × 75'

Moffettt Field NAS

ID: NUQ City: Moffett, CA
Elevation: 34' Owner: US Navy
Rwy: 14L - 32R: 9200' × 200' Rwy: 14R - 32L: 8120' × 200'

Napa County

ID: APC City: Napa, CA
Elevation: 33' Owner: Napa County
Rwy: 18R - 36L: 5931' × 150' Rwy: 06 - 24: 5007' × 150'
Rwy: 18L - 36R: 2500' × 75'

Palo Alto

ID: PAO | City: Palo Alto, CA
Elevation: 5' | Owner: County of Santa Clara
Rwy: 12 - 30: 2500' × 65'

Pataluma Muni

ID: O69 | City: Petaluma, CA
Elevation: 87' | Owner: City of Petaluma
Rwy: 11 - 29: 3600' × 75'

Reid-Hillview

ID: RHV | City: San Jose, CA
Elevation: 133' | Owner: County of Santa Clara
Rwy: 13L - 31R: 3101' × 75' | Rwy: 13R - 31L: 3099' × 75'

San Carlos

ID: SQL | City: San Carlos, CA
Elevation: 2' | Owner: San Mateo County
Rwy: 12 - 30: 2600' × 75'

San Francisco IntÕl.

ID: SFO | City: San Francisco, CA
Elevation: 11' | Owner: Cty & Co of SF
Rwy: 01R - 19L: 8901' × 200' | Rwy: 01L - 19R: 7001' × 200'
Rwy: 10L - 28R: 11,870' × 200' | Rwy: 10R - 28L: 10,600' × 200'

San Jose IntÕl.

ID: SJC | City: San Jose, CA
Elevation: 56' | Owner: City of San Jose
Rwy: 12R - 30L: 8899' × 150' | Rwy: 11 - 29: 4599' × 100'
Rwy: 12L - 30R: 4419' × 150'

Sonoma Skypark

ID: 0Q9 | City: Sonoma, CA
Elevation: 20' | Owner: Sonoma Skypark
Rwy: 08 - 26: 2500' × 30'

Sonoma Valley

ID: 0Q3 | City: Schellville/Sonoma, CA
Elevation: 11' | Owner: W. Reichelt
Rwy: 07 - 25: 2700' × 35' | Rwy: 16 - 34: 2200' × 65'

Hawaiian Islands Airports

Figure 4 shows the Hawaiian islands. The airports in the Hawaiian Islands are shown and described on the following pages.

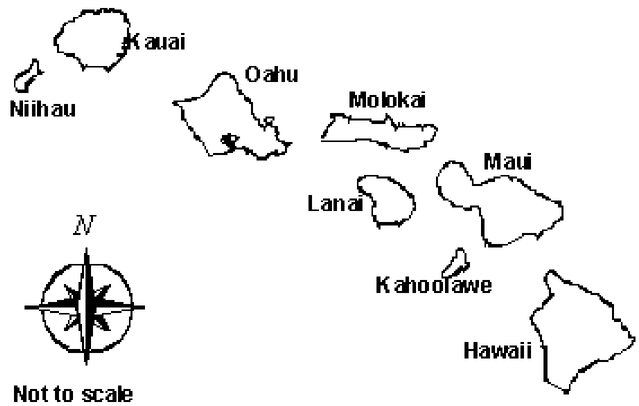


Figure 4 Hawaiian Islands

Kauai Airports

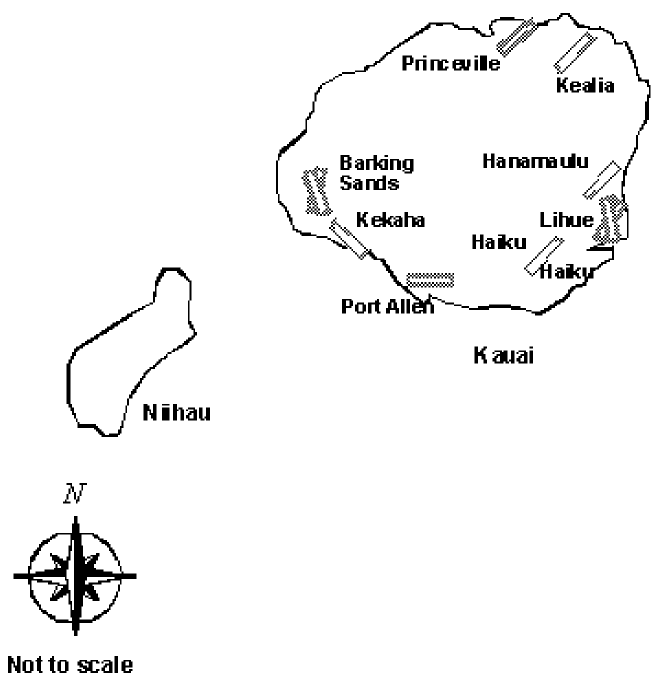


Figure 5 Kauai Airports

Barking Sands PMRF

ID: BKH
City: Kekaha, HI
Elevation: 14'
Owner: US Navy
Rwy: 01 - 19: 6000' × 150'
Rwy: 16 - 34: 6000' × 150'

Haiku Airstrip

ID: HI33
City: Puhi, HI
Elevation: 3854'
Owner: McBryde Sugar Co
Rwy: NE - SW: 2400' × 60'

Hanamaulu Airstrip

ID: HI03
City: Hanamaulu, HI
Elevation: 404'
Owner: Lihue Plantation Co
Rwy: NE - SW: 1600' × 50'

Kealia Airstrip

ID: HI15
City: Kealia, HI
Elevation: 551'
Owner: Lihue Plantation Co
Rwy: NE - SW: 1940' × 60'

Kekaha Airstrip

ID: HI16
City: Kekaha, HI
Elevation: 6'
Owner: Kekaha Sugar Co
Rwy: 09 - 27: 2450' × 60'

Lihue

ID: LIH
City: Lihue, HI
Elevation: 149'
Owner: Hawaii St Arpts Div
Rwy: 03 - 21: 6000' × 100'
Rwy: 17 - 35: 6500' × 150'
Rwy: H1: 64' × 64'

Port Allen

ID: PAK
City: Hanapepe, HI
Elevation: 24'
Owner: Hawaii St Arpts Div
Rwy: NW - SE: 1500' × 40'

Princeville

ID: HI01
City: Hanalei, HI
Elevation: 320'
Owner: Princeville Develop
Rwy: 05 - 23: 3380' × 40'

Oahu Airports

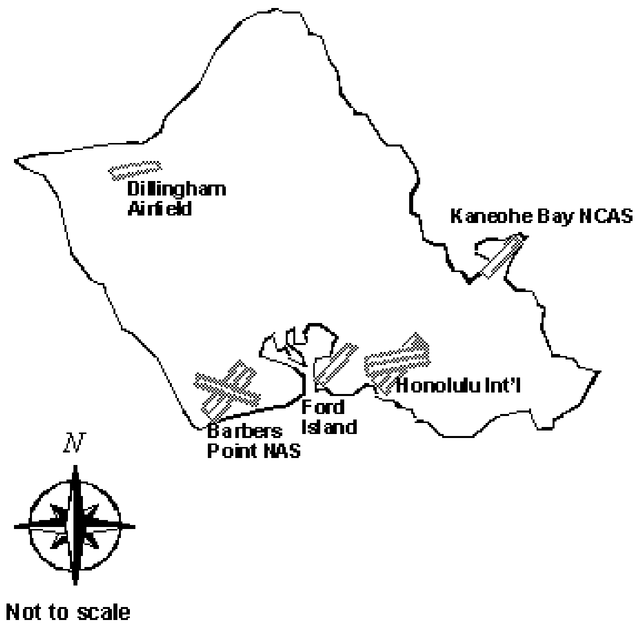


Figure 6 Oahu Airports

Barbers Point NAS (John Rodgers Field)

ID: NAX	City: Ewa, HI
Elevation: 34'	Owner: US Navy
Rwy: 04L - 22R: 8330' × 200'	Rwy: 04R - 22L: 8330' × 200'
Rwy: 11 - 29: 8411' × 200'	

Dillingham Airfield

ID: HDH	City: Mokuleia, HI
Elevation: 15'	Owner: US Army
Rwy: 08 - 26: 9000' × 75'	

Ford Island ALF

ID: NPS	City: Honolulu, HI
Elevation: 18'	Owner: US Navy
Rwy: 04 - 22: 4000' × 50'	

Honolulu IntÕl

ID: HNL	City: Honolulu, HI
Elevation: 13'	Owner: Hawaii St & USAF
Rwy: 04L - 22R: 6953' × 150'	Rwy: 04R - 22L: 9000' × 150'
Rwy: 08L - 26R: 12,357' × 150'	Rwy: 08R - 26L: 12,000' × 200'
Rwy: H1: 100' × 100'	

Kaneohe Bay MCAS (Mokapu Point)

ID: NGF	City: Kaneohe, HI
Elevation: 18'	Owner: US Navy
Rwy: 04 - 22: 7767' × 200'	

Molokai Airports

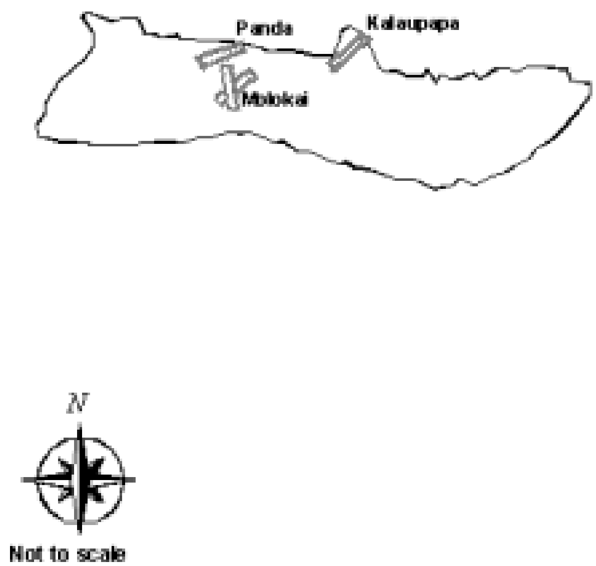


Figure 7 Molokai Airports

Kalaupapa

ID: LUP	City: Kalaupapa, HI
Elevation: 26'	Owner: Hawaii St Airports Div
Rwy: 05 - 23: 2760' × 50'	Rwy: 17 - 35: 3118' × 100'

Molokai

ID: MKK	City: Kaunakakai, HI
Elevation: 454'	Owner: Hawaii St Airports Div
Rwy: 05 - 23: 4494' × 100'	Rwy: 17 - 35: 3118' × 100'

Panda

ID: HI49	City: Kaunakakai, HI
Elevation: 250'	Owner: John Hutton Corp
Rwy: 07 - 25: 1920' × 55'	

Lanai Airports

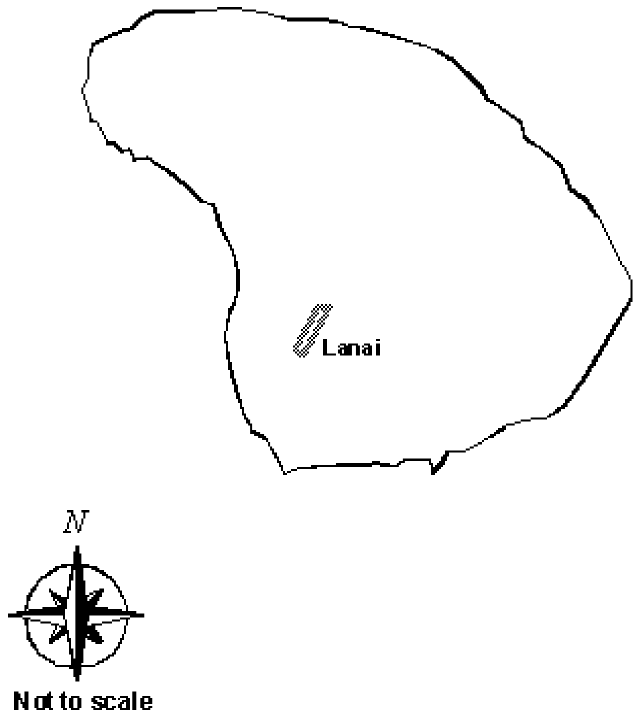


Figure 8 Lanai Airports

Lanai

ID: LNY
City: Lanai City, HI
Elevation: 1308'
Owner: Hawaii St Airports Div
Rwy: 03 - 21: 5000' × 150'

Maui Airports

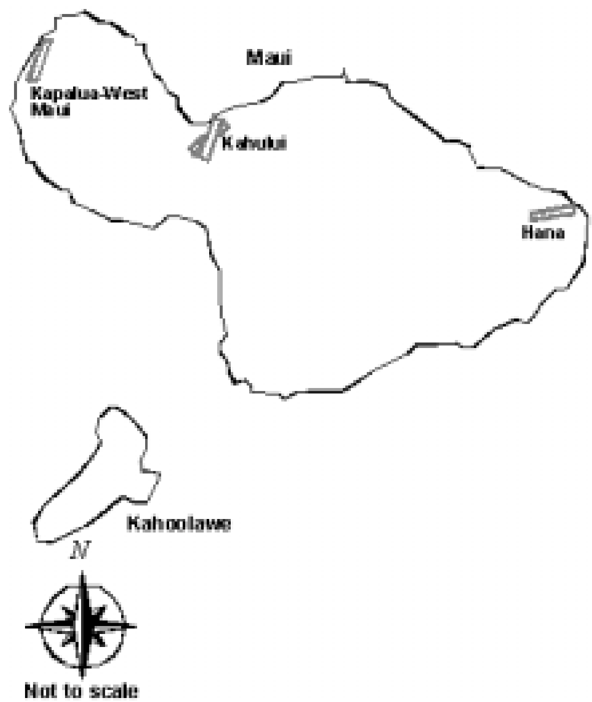


Figure 9Maui and Kahoolawe Airports

Hana

ID: HNM
City: Hana, HI
Elevation: 77'
Owner: Hawaii St Airports Div
Rwy: 08 - 26: 3605' × 100'

Kahului (Maui)

ID: OGG
City: Kahului, HI
Elevation: 54'
Owner: Hawaii St Airports Div
Rwy: 02 - 20: 6995' × 150'
Rwy: 05 - 23: 4990' × 150'
Rwy: H1: 75' × 75'

Kapalua-West Maui

ID: JHM
City: Lahaina, HI
Elevation: 249'
Owner: West Maui Airport Inc
Rwy: 02 - 20: 3000' × 100'

Hawaii Airports

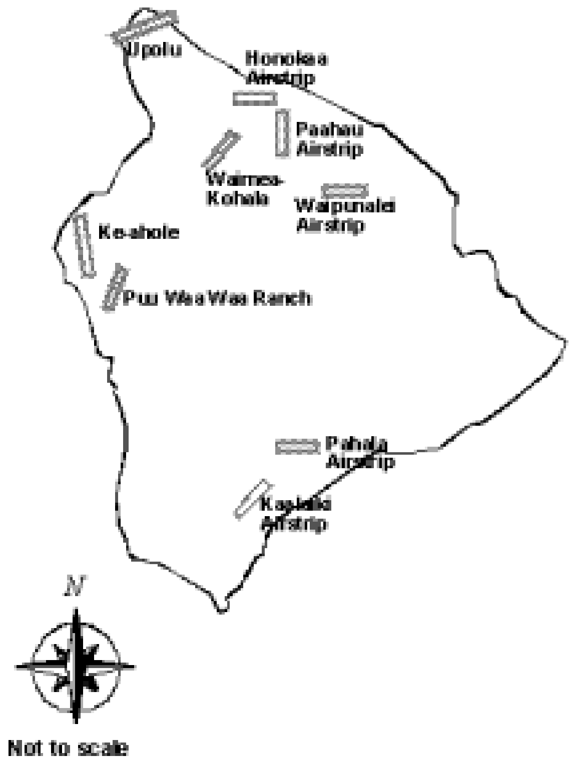


Figure 10 Hawaii Airports

Bradshaw Army Airfield

ID: BSF
City: Camp Pohakuloa, HI
Elevation: 6125'
Owner: USAR HAW
Rwy: 09 - 27: 3700' × 90'

Hilo International

ID: ITO
City: Hilo, HI
Elevation: 38'
Owner: Hawaii St Airports Div
Rwy: 03 - 21: 5600' × 150'
Rwy: 08 - 26: 9800' × 150'

Honokaa Airstrip

ID: HI05
City: Honokaa, HI
Elevation: 1440'
Owner: Hamakua Sugar Co
Rwy: E - W: 1800' × 35'

Kaalaiki Airstrip

ID: HI25
City: Naalehu, HI
Elevation: 1964'
Owner: KaÕu Agribusiness Co
Rwy: NE - SW: 2000' × 40'

Ke-ahole

ID: KOA
City: Kailua/Kona, HI

Elevation: 43'
Owner: Hawaii St Airports Div
Rwy: 17 - 35: 6500' × 150'

Mauna Kea-Honolii

ID: HI31
City: Papaikou, HI
Elevation: 1400'
Owner: Mauna Kea Agri Co
Rwy: 07 - 25: 1800' × 25'

Mountain View Airstrip

ID: HI23
City: Mountain View, HI
Elevation: 1500'
Owner: Keaau Agri Land Corp
Rwy: 05 - 23: 1800' × 40'

Paauhau Airstrip

ID: HI06
City: Hokokaa, HI
Elevation: 2060'
Owner: Hamakua Sugar Co
Rwy: N - S: 1657' × 22'

Pahala Airstrip

ID: HI28
City: Pahala, HI
Elevation: 1195'
Owner: IntÕl Air Svc Co
Rwy: E - W: 1650' × 70'

Peleau

ID: HI02
City: Hakalau, HI
Elevation: 1088'
Owner: Mauna Kea Agri
Rwy: 07 - 25: 2100' × 25'

Pepeekeo Airstrip

ID: HI32
City: Pepeekeo, HI
Elevation: 675'
Owner: Mauna Kea Agri Co
Rwy: NE - SW: 2000' × 25'

Puu Waa Waa Ranch

ID: HI13
City: Kailua/Kona, HI
Elevation: 2250'
Owner: F. Newell Bohnett
Rwy: 02 - 20: 2950' × 40'

Upolu

ID: UPP
City: Hawi, HI
Elevation: 96'
Owner: Hawaii St Airports Div
Rwy: 07 - 25: 3800' × 75'

Upper Paauai

ID: HI29
City: Pahala, HI
Elevation: 2600'
Owner: KaÕu Agribusiness Co
Rwy: NW - SE: 1965' × 40'

Upper Paauilo Airstrip

ID: HI13
City: Paauilo, HI

Elevation: 1520' Owner: Hamakua Sugar Co
Rwy: NE - SW: 1400' × 36'

Waimea-Kohala

ID: MUE City: Kamuela, HI
Elevation: 2671' Owner: Hawaii St Airports Div
Rwy: 04 - 22: 5197' × 100'

Waipunalei Airstrip

ID: HI20 City: Laupahoehoe, HI
Elevation: 1360' Owner: Hamakua Sugar Co
Rwy: 09 - 27: 2100' × 50'

D. AVDS Terrain Library Functions

(ATL_COLOR, ATL_COLOR_POST, ATL_ELEVATION, ATL_ELEVATION_POST, ATL_PCLEVELS, ATL_RASTERFILE, ATL_SHADING)

ATL_COLOR

Generates colors in AVDS terrain data files.

int atl_color(double **long_00,** double **lat_00,** double **long_0c,** double **lat_0c,** double **long_r0,** double **lat_r0,** double double **long_rc,** double **lat_rc,** char* **filename,** int **num_rows,** int **num_cols,** double **gamma)**

RETURN VALUES

Returns an integer with the value 0 on success and 1 on failure.

REQUIRED HEADER

#include <atl.h>

PARAMETERS

long_00, lat_00 represent the longitude and latitude of the location on the globe of the color[0][0].,
long_0c, lat_0c indicate the location the color[0][num_cols-1].
long_r0, lat_r0 indicate the location the color[num_rows-1][0]
long_rc, lat_rc indicate the location the color[num_rows-1][num_cols-1]
NOTE: All longitudes are measured in degrees from the Meridian of Greenwich and all latitudes are measured in degrees from the Equator.
Filename points to the pathname of a file containing the colors that are to be converted. The file must contain the color data in the format of arrayed atl_color_struct structure:

```
      struct atl_struct_color color[num_rows][num_cols];
```

The file usually contains the colors from digital data (Landsat) or a rasterized photograph that is to be used to color the terrain. Each color contains a red, green, and blue color component in the range from 0 (black) to 255 (full intensity). Each color also contains an alpha component in the range from 0 (color not used) to 255 (color fully used), which indicates how the color should be blended into the database. The atl_struct_color structure is defined in <atl.h> as follows:

```
      struct atl_struct_color {
                  unsigned char alpha;
                  unsigned char blue;
                  unsigned char green;
                  unsigned char red;
                  };
```

num_rows, num_cols number of rows and columns in the color array
gamma Sometimes raster images are gamma corrected by an amount so that they appear correct when displayed on a monitor. The colors are converted by atl_color() from gamma corrected space to linear color space where calculations are done. If the image is already in linear color space, gamma should be set to 1.0. If the image has been gamma corrected (eg. 2.22), gamma should be set to the inverse of the gamma correction used (eg. 0.45).

REMARKS

atl_color() changes or creates the colors in the data files used by AVDS. The intermediate data files have filenames of ???_???.c32 where ? is a number. atl_color() allows the colors in the data files to be handled in small sections. atl_shading() then can be run on all the data. When all the colors have been generated, atl_color_post() must be run on all of the data.

SEE ALSO

atl_color_post(), atl_shading()

ATL_COLOR_POST

Final color processing of AVDS terrain data files.

int atl_color_post(double long_00, double lat_00, double long_0c, double lat_0c, double long_r0, double lat_r0, double long_rc, double lat_rc, double gamma, int flags)

RETURN VALUES

Returns an integer with the value 0 on success and 1 on failure.

REQUIRED HEADER

```
#include <atl.h>
```

PARAMETERS

long_00, lat_00, long_0c, lat_0c, long_r0, lat_r0, long_rc, lat_rc represent the longitudes and latitudes of the locations on the globe of the area bounding box where the color postprocessing is to be done. All longitudes are measured in degrees from the Meridian of Greenwich and all latitudes are measured in degrees from the Equator.

Gamma For correct colors to appear on the monitor, gamma should be set to the gamma correction of the monitor (eg. 2.50)..

flags atl_color_post() can generate colors for different depth frame buffers. flags values are constructed by ORing flags from the following list (only one of the first two flags below must be used):

- ATL_DEPTH_4 Create colors for 4 bit color map double buffering.
- ATL_DEPTH_8 Create colors for 8 bit frame buffers.
- ATL_BLUE_WATER Use dark blue whenever the elevation is zero.

REMARKS

atl_color_post()converts from linear color space to gamma corrected space for ATL_DEPTH_4 and ATL_DEPTH_8. atl_color_post() does the color postprocessing needed by the files used by AVDS. Final colors and color maps are calculated. The intermediate input color data files have filenames of ???_???.c32 and the output filenames are ???_??? where ? is a number.

atl_color_post() must be called once after the data files have been generated by one or more calls to atl_color(), atl_color_delete(), and atl_shading().

SEE ALSO

atl_color(), atl_color_delete(), atl_shading()

ATL_ELEVATION

Generate elevations in AVDS terrain data files.

int atl_elevation(double **long_00,** double **lat_00,** double **long_0c,** double **lat_0c,** double **long_r0,** double **lat_r0,** double **long_rc,** double **lat_rc,** char* **filename,** int **num_rows,** int **num_cols)**

RETURN VALUES

Returns an integer with the value 0 on success and 1 on failure.

REQUIRED HEADER

```
#include <atl.h>
```

PARAMETERS

long_00, lat_00 represent the longitude and latitude of the location on the globe of the elevation[0][0].
long_0c, lat_0c indicate the location the elevation[0][num_cols-1].
long_00, lat_r0 indicate the location the elevation[num_rows-1][0].
long_rc, lat_rc indicate the location the elevation[num_rows-1][num_cols-1].
Note: All longitudes are measured in degrees from the Meridian of Greenwich and all latitudes are measured in degrees from the Equator.
filename filename points to the pathname of a file containing the elevations that are to be converted. The file must contain the elevation data in the format of an array of integers:

```
      int elevation[num_rows][num_cols];
```

The array, elevation[num_rows][num_cols], contains the heights measured in millimeters from sea level.
num_rows, num_cols Number of rows and columns of elevations in the file.

REMARKS

atl_elevation() changes or creates the elevations in the data files used by AVDS. The data files have filenames of ???_??? where ? is a number. atl_elevation() allows the elevations of the data files to be handled in small sections. When all the elevations have been generated, atl_elevation_post() must be run on all of the data.

SEE ALSO

atl_elevation_post()

ATL_ELEVATION_POST

Final elevation processing of AVDS terrain data files.

int atl_elevation_post(**double** long_00, **double** lat_00, **double** long_0c, **double** lat_0c, **double** long_r0, **double** lat_r0, **double** long_rc, **double** lat_rc)

RETURN VALUES

Returns an integer with the value 0 on success and 1 on failure.

REQUIRED HEADER

```
#include <atl.h>
```

PARAMETERS

long_00, lat_00, long_0c, lat_0c; long_r0, lat_r0, long_rc, lat_rc
long_00 and lat_00, long_0c and lat_0c, long_r0 and lat_r0, and long_0r and lat_0r represent the longitudes and latitudes of the locations on the globe of the area bounding box where the elevation postprocessing is to be done. All longitudes are measured in degrees from the Meridian of Greenwich and all latitudes are measured in degrees from the Equator.

REMARKS

atl_elevation_post() does the elevation postprocessing needed by the data files used by AVDS. Minimum and maximum elevations are calculated. The data files have filenames of ???_??? where ? is a number. atl_elevation_post() must be called once after the data files have been generated by one or more calls to atl_elevation() and atl_elevation_delete().

SEE ALSO

atl_elevation()

ATL_PCLEVELS

Generate elevations in AVDS terrain data files.

> **int atl_pclevels (**double **long_00,** double **lat_00,** double **long_0c,** double **lat_0c,** double **long_r0,** double **lat_r0,** double **long_rc)**

RETURN VALUES

Returns an integer with the value 0 on success and 1 on failure.

REQUIRED HEADER

```
#include <atl.h>
```

PARAMETERS

long_00, lat_00, long_0c, lat_0c; long_r0, lat_r0, long_rc, lat_rc

long_00 and lat_00, long_0c and lat_0c, long_r0 and lat_r0, and long_0r and lat_0r represent the longitudes and latitudes of the locations on the globe of the area bounding box where the processing is to be done. All longitudes are measured in degrees from the Meridian of Greenwich and all latitudes are measured in degrees from the Equator.

REMARKS

atl_pclevels() reads in the multiple detail level AVDS terrain files inside of the specified boundaries and outputs multiple reduced detail AVDS terrain files. Output files have the form ???_???_? where the ?'s are numbers.

SEE ALSO

ATL_RASTERFILE

Create a windows bitmap file from AVDS terrain data files.

int atl_rasterfile(double long_00**, double** lat_00**, double** long_0c, **double** lat_0c, **double** long_r0, **double** lat_r0, **double** long_rc, **double** lat_rc, **char*** filename, **int** flags**)**

RETURN VALUES

Returns an integer with the value 0 on success and 1 on failure.

REQUIRED HEADER

```
#include <atl.h>
```

PARAMETERS

long_00, lat_00, long_0c, lat_0c; long_r0, lat_r0, long_rc, lat_rc represent the longitudes and latitudes of the locations on the globe of the the four corners of the bitmap file. All longitudes are measured in degrees from the Meridian of Greenwich and all latitudes are measured in degrees from the Equator.

Filename atl_rasterfile() creates a bitmap file pointed to by filename from AVDS terrain data files

flags atl_rasterfile() can generate a bitmap file from the different depth colors in the terrain data files. flags specifies the depth that will be used to generate the bitmap file (only one of the flags below must be used):

ATL_DEPTH_4 Create rasterfile using 4 bit color data.
ATL_DEPTH_8 Create rasterfile using 8 bit color data.

REMARKS

The bitmap file can be used as an aid during the development of the terrain data. atl_rasterfile() creates the bitmap file directly from the color data in the files ???_???, without modifying the colors in any way.

atl_rasterfile() can only be called after the data files have been generated and postprocessed using atl_color_post() and atl_elevation_post().

SEE ALSO

atl_color_post(), atl_elevation_post()

ATL_SHADING

Generate shading in AVDS terrain data files.

int atl_shading(**double** long_00, **double** lat_00, **double** long_0c, **double** lat_0c, **double** long_r0, **double** lat_r0, **double** long_rc, **double** lat_rc, **struct atl_color_struct***color, **double** time_of_day)

RETURN VALUES

Returns an integer with the value 0 on success and 1 on failure.

REQUIRED HEADER

```
#include <atl.h>
```

PARAMETERS

long_00, lat_00, long_0c, lat_0c; long_r0, lat_r0, long_rc, lat_rc
long_00 and lat_00, long_0c and lat_0c, long_r0 and lat_r0, and long_rc and lat_rc represent the longitudes and latitudes of the locations on the globe of the area bounding box where the shading is to be done. All longitudes are measured in degrees from the Meridian of Greenwich and all latitudes are measured in degrees from the Equator.
**color* Pointer to an array of atl_struct_color with two elements which define the dark color, color[0], and light color, color[1].
time_of_day . The sun's location is described by time_of_day in military time (eg. 16.25 = 4:15pm).

REMARKS

atl_shading() changes or creates the shading of the terrain in the data files used by AVDS. Shading gives visual details about the slopes of the terrain. The input elevation data files have filenames of ???_??? and the intermediate input/output color data files have filenames of ???_???.c32 where ? is a number. atl_shading() should be used when no shaded color data of the terrain is available or when color data needs to be shaded. atl_shading() uses diffused reflection shading with the light source being the sun
If no color information is available for a location in the data file, a diffused color is generated from the dark color[0] and light color[1]. color[] is specified in linear color space.
atl_shading() can also be used to shade nonshaded color data of the terrain. If there is color information for a location then normal diffused reflection is done on that color. color[0] and color[1] are ignored.
atl_shading() should be called once after the data files have been generated by one or more calls to atl_elevation() and/or atl_color() and before atl_color_post().

SEE ALSO

atl_color(), atl_color_post(), atl_elevation()

E. DXF to AVDS Conversion utility

GENERAL

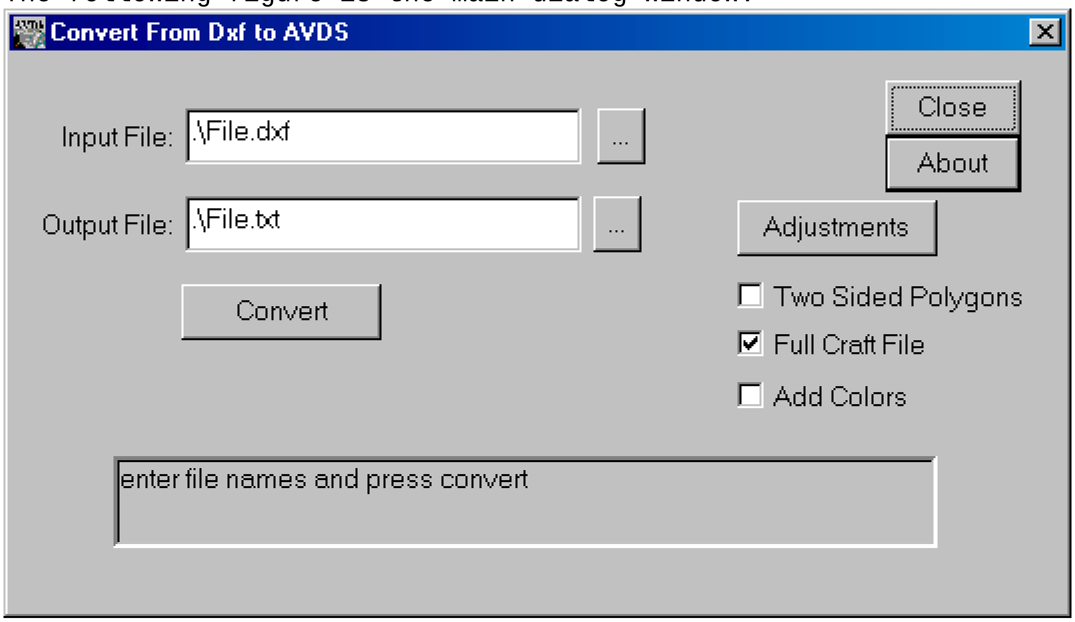
This utility is a windows based program that converts 3D graphics files from Autodesk's AutoCAD® DXF format to the AVDS craft file format. The only 3D DXF elements supported are the 3DFACE and POLYLINE elements.

ADDING NEW AVDS CRAFT FILES

To add a new craft to AVDS craft

E.3.1 USER INTERFACE

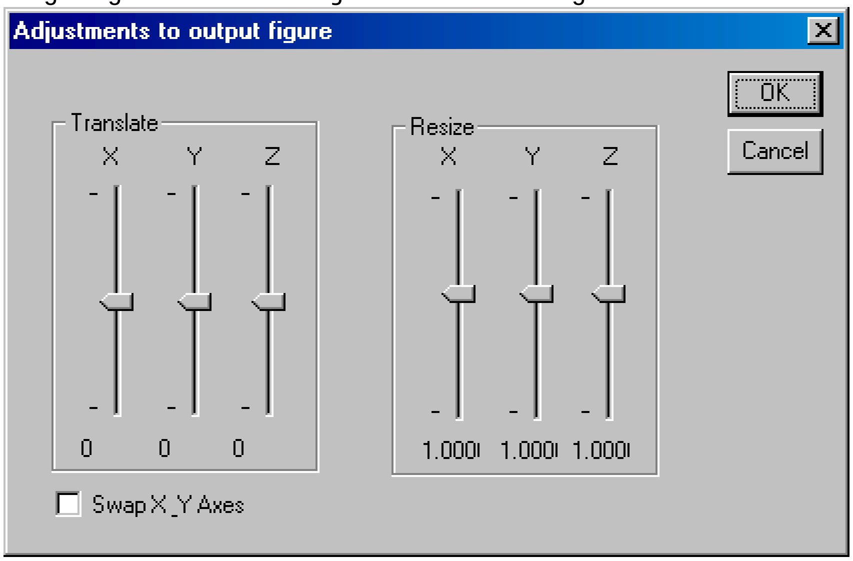
The following figure is the main dialog window:



Input File: this is the entry blank for the file containing the DXF data. Note: pressing the button , will bring up a dialog a file open window to aid in file selection.
Output File: this is the entry blank for the file that will contain the resulting AVDS craft data. Note: pressing the button , will bring up a dialog a file open window to aid in file selection.
Two Sided Polygons: Checking this block causes two polygons to be drawn for each DXF 3DFACE, each one visible from one side. This feature is used when some of the 3DFACE polygons are not facing in the direction required to make them visible.
Full Craft File: Checking this block causes all of the extra elements necessary for an AVDS craft file to be added to the output file. If this block is not checked, the user will need to add the contents of the output file to another file that contains the extra craft information.
Add Colors: Checking this block causes the conversion utility to add color statements into the craft after every group of entities. This marks the transitions between blocks. This is most useful when there are no colors define in the DXF file.
Convert: Pressing this button causes the conversion to take place.

Adjustments: Pressing this button causes the "Adjustments" dialog window, see below, to open.

The following figure is the Adjustments dialog window:



The adjustments made in this window will be applied to the converted out figure during the conversion process.
Translate: Adjustment of these three sliders causes the output figure to be moved linearly (translated) in the three axes.
Resize: Adjustment of these three sliders causes the output figure to be scaled (resized) in the three axes.

F. UserCodeBlock

General

The UserCodeBlock (UCB), introduced in version 1.3, is a programming interface that makes it possible for users to create and use programs that are integrated with AVDS. With a UCB a user can add do things such as add multiple vehicles during interactive simulation, draw their own HUD or radar display, in the AVDS window, during playback or interactive simulation. UCB are written in C++ and are based on the class, “CCodeBlockUser”. The C++ code is compiled into a dynamic-link library (DLL) which is loaded when AVDS is first started. As explained below, the built-in functions in the UCB enable the user to draw 2D and 3D objects in the AVDS environment, run dynamic simulations, and control built-in AVDS vehicles, including movable surfaces such as elevators and ailerons. There are four examples of the user of UCBs, *UserHUD*, *UserRadar*, *UserWingman,* and *UserText*. These can be found in the directory “AVDS\model” and are documented in appendix **G**. *UserHUD* implements a heads-up display in the UCB. *UserRadar* implements a radar display in a UCB. *UserWingman* implements a second vehicle in a UCB. *UserText* implements the ability to add titles to AVDS.

CCodeBlockUser

This is the base class for all UserCodeBlocks. The UCB must inherit from CCodeBlockUser. This makes it possible for AVDS to control the execution of the UCB. The member functions of CCodeBlockUser represent the connections between the UCB and AVDS.

Required Header

```
#include < CodeBlockUser.h>
```

Dynamic-Link Library (DLL) Construction

There are two symbols required as exports from the UCB DLL. These are: ***cbVersion*** and ***cbUser***. Both of these symbols must be defined as globals symbols. ***cbVersion*** is the version number of the ***CodeBlockUser.h*** file. ***cbUser*** is an instance of the UCB class. The following is a typical declaration:

```
extern "C"
{
	double cbVersion = VERSION_USER_CODE_BLOCK;
	CUserHUD cbUser;
}
```

Note: *VERSION_USER_CODE_BLOCK* is defined in ***CodeBlockUser.h***.

Variable Types

Some of the variable types used were constructed using macro definitions or typedef. The following is a list of those variables and their built-in types.

BOOL *– int*
V_DOUBLE_t - vector<double>
V_DOUBLE_IT_t *- V_DOUBLE_t::iterator*
V_BOOL_t *- vector<BOOL>*
V_BOOL_IT_t *- V_BOOL_t::iterator V_BOOL_IT_t*

Common UCB Member Function Parameters

The UCBs have a number of virtual member functions that users can overide with their own code. AVDS calls these functions at the appropriate times. Many of the member functions use the same structure. For example, the following is the function prototype for the update visual function:

```
	virtual BOOL bUpdateVisual(double dTimeElapsed,
			const double*& pdInputs,size_t szNumberInputs,
			double*& pdOutputs,size_t szNumberOutputs,
			stringstream& ssError)
```

Return Values

The member functions that return Boolean, BOOL, values must return either TRUE or FALSE. ***TRUE*** indicates that the function was **completed successfully**. A return value of ***FALSE*** indicates that the function was **not completed successfully** and will cause AVDS to terminate the simulation or playback and display a dialog window with the returned string, see the entry for ***ssError*** below.

Parameters

double ***dTimeElapsed*** - the time, in seconds that has elapsed since the ***START*** button was pressed.
const double&* ***pdInputs*** - the array of inputs that were connected to the block using the Simulation Configuration or Playback Blocks configuration dialog, see below. For example:

```
double dInputOfInterest = pdInputs[2];
```

size_t ***szNumberInputs*** – the number of inputs in the ***pdInputs*** array.
double&* ***pdOuputs*** - the array of outputs from the UCB. For example:

```
pdOutputs[2] = dInputOfInterest;
```

size_t ***szNumberOuputs*** – the number of outputs in the ***pdOuputs*** array.
stringstream ***ssError*** – the container for returning an error message. When ***FALSE*** is returned from the function, AVDS displays any message contained in ***ssError*** in a pop-up dialog window. For example:

```
ssError << “An error occurred in UCBXYZ”;
return(FALSE);
```

UCB Member Functions

virtual CCodeBlockUser* cbuClone(stringstream& ssError)

cbuClone is used by AVDS to creat new instances of the UCB. The user should override this function and return a pointer to a new instance of the UCB. For example:

```
CCodeBlockUser* CUserHUD::cbuClone(stringstream& ssError)
{
      return(new CUserHUD());
}
```

stringstream ***ssError*** – the container for returning an error message. AVDS will signal an error is ***ssError*** is returned not empty. For example:

```
CCodeBlockUser* CUserHUD::cbuClone(stringstream& ssError)
{
      CCodeBlockUser* pucbTemp = new CUserHUD();
      if(!pucbTemp)
            ssError << “Error creating CCodeBlockUser”;
      return(pucbTemp);
}
return(FALSE);
```

BOOL bConfigure(HWND hWndParent,stringstream& ssError)

bConfigure is called when a user *right-clicks* on the entry for the UCB in the Simulation Configuration or Playback Blocks Configuration dialog windows. Users can use this function to bring up a user-defined window to configure the UCB. For a simple example of the use of this function, see the example UCB ***UserRadar***. For a more complicated example, see ***UserHUD***. Both of these examples can be found in the directory “AVDS\model”.

BOOL bConfigurationFileWrite(const char* cstrSection,const char* cstrConfigFile)

bConfigurationFileWrite is called when a user presses the *Save* or *SaveAs* buttons on the Simulation Configuration or Playback Blocks Configuration dialog windows. Users can use this function to save the configuration of the UCB. For a simple example of the use of this function, see the example UCB ***UserText***. This example can be found in the directory “AVDS\model\UserText”.

BOOL bConfigurationFileRead(const char* cstrSection,const char* cstrConfigFile,BOOL bReload)

bConfigurationFileRead is called at the appropriate times for the UCB to load any saved configuration data. Users can use this function to load the configuration of the UCB. For a simple example of the use of this function, see the example UCB ***UserText***. This example can be found in the directory "AVDS\model\UserText".

```
BOOL bUpdateVisual (double dTimeElapsed,
        const double*& pdInputs,size_t szNumberInputs,
        double*& pdOutputs,size_t szNumberOutputs,
        stringstream& ssError)
```

bUpdateVisual can be used to draw 3D objects in the AVDS environment. OpenGL commands can be used to draw these 3D elements. This function is called before AVDS redraws the graphics window. For definition of the parameters see the *Common UCB Member Function Parameters* section.

```
BOOL bDraw2D (double dTimeElapsed,
        const double*& pdInputs,size_t szNumberInputs,
        double*& pdOutputs,size_t szNumberOutputs,
        stringstream& ssError)
```

bDraw2D can be used to draw 2D objects in the AVDS environment. OpenGL commands can be used to draw these 2D elements. This function is called before AVDS redraws the graphics window. For definition of the parameters see the *Common UCB Member Function Parameters* section.

BOOL *bInitialize* (double dTimeElapsed,
const double*& pdInputs,size_t szNumberInputs,
double*& pdOutputs,size_t szNumberOutputs,
stringstream& ssError)

bInitialize is used to performs tasks at the start of the simulation or playback run. It is called when the ***START*** button is pressed before the simulation or playback starts. For definition of the parameters see the *Common UCB Member Function Parameters* section.

```
BOOL bUpdateDynamics (double dTimeElapsed,
        const double*& pdInputs,size_t szNumberInputs,
        double*& pdOutputs,size_t szNumberOutputs,
        stringstream& ssError)
```

bUpdateDynamics is used to update dynamic simulation models during interactive simulation or playback. During the run, this function is called at aproximately the rate specified in the member storage variable, ***m_dUpdateRate_hz***. For definition of the parameters see the *Common UCB Member Function Parameters* section.

```
BOOL bExitCleanUp (double dTimeElapsed,
        const double*& pdInputs,size_t szNumberInputs,
        double*& pdOutputs,size_t szNumberOutputs,
        stringstream& ssError)
```

bExitCleanUp is used to perform any tasks necessary at the end of the simulation. This function is called when the ***STOP*** button is pressed. For definition of the parameters see the *Common UCB Member Function Parameters* section.

UCB Member Storage

There are many storage members defined in the class. AVDS uses these class members to configure the UCB. Each of the storage member describe below has associated accessor functions, i.e the accessors for the member ***m_strBlockLabel*** are:

```
string& strGetBlockLabel(){return(m_strBlockLabel);};
const string& strGetBlockLabel()const{return(m_strBlockLabel);};
```

The accessor functions should be used in the place of the storage members, i.e:

```
strGetBlockLabel() = "This is UCB XYZ";
string  strTemp = strGetBlockLabel();
```

string m_strBlockLabel

This string is used to display the block label in the Simulation Configuration or Playback Blocks Configuration dialog windows.

string m_strInputs

This string is used to define the names and numbers of *inputs* to the UCB. ***m_strInputs*** must contain a list of *input* names separated by spaces or tabs.

string m_strOutputs

This string is used to define the names and numbers of *outputs* from the UCB. ***m_strOutputs*** must contain a list of *output* names separated by spaces or tabs.

double m_dUpdateRate_hz

This is the target rate, in Hz, for AVDS to call the ***bUpdateDynamics*** function.

int m_iGeometryIndex

AVDS uses this member storage to determine if to associate a vehicle geometry with the UCB. It also determines which geometry to display. If ***bUpdateDynamics*** is negative, then no craft will be associated with this UCB. If ***bUpdateDynamics*** if greater than or equal to zero, then its value is used to select the vehicle geometry from the list of vehicles in the file "AVDS\craft\craftcap.txt". Note: the numbering of the craft in the list starts at zero.

double m_dInitLatitude_degd

If a vehicle geometry is associated with this UCB, then this member is used to set the initial latitude. The latitude is given in degrees decimal.

double m_dInitLongitude_degd

If a vehicle geometry is associated with this UCB, then this member is used to set the initial longitude. The longitude is given in degrees decimal.

BOOL m_bUseModelingMatrix

If this member is set to TRUE, AVDS uses the values in m_vdTranslate and m_vdRotate to create and implements a modeling matrix in OpenGL. With this modeling matrix active the user can place use OpenGL commands in the *bUpdateVisual* function to draw 3-dimensional objects in the environment.

V_DOUBLE_t m_vdTranslate

If a vehicle geometry is associated with this UCB, then this member is used to set the linear position of the vehicle relative to the initial latitude and longitude. The positions are x, y, and z in feet, i.e.:

```
vdGetTranslate()[0] = 26.3;   //North
vdGetTranslate()[1] = 26.3;   //East
vdGetTranslate()[2] = 26.3;   //Down
```

V_DOUBLE_t m_vdRotate

If a vehicle geometry is associated with this UCB, then this member is used to set rotations of the vehicle about the body axes. The rotations are x, y, and z in degrees, i.e.:

```
vdGetRotate()[0] = 26.3;      //Phi
vdGetRotate ()[1] = 26.3;     //Theta
vdGetRotate ()[2] = 26.3;     //Psi
```

V_DOUBLE_t m_vdSurfaceRotation

If a vehicle geometry is associated with this UCB, then this member is used to set rotations of the vehicle's articulated surfaces. The rotations are in radians and the possible surfaces are enumerated in the enum type **enCraft**, see the file "CodeBlockUser.h", i.e.:

```
vdGetSurfaceRotation()[craftArticulatedSurface01] = 0.01;
```

V_DOUBLE_t m_vdTransparency

If a vehicle geometry is associated with this UCB, then this member is used to set the transparency level of the of the vehicle's category of polygons. The transparency ranges from 1.0 to 0.0 and possible categories are enumerated in the enum type **enCraft**, see the file "CodeBlockUser.h", i.e.:

```
vdGetTransparency()[craftExhaust] = 0.1;
```

V_DOUBLE_t m_vdColorDensity

If a vehicle geometry is associated with this UCB, then this member is used to set the color density level of the of the vehicle's category of polygons. The color density ranges from 0.0 to 1.0 and possible categories are enumerated in the enum type **enCraft**, see the file "CodeBlockUser.h", i.e.:

```
vdGetColorDensity()[craftExhaust] = 0.1;
```

V_BOOL_t m_vbShowComponent

If a vehicle geometry is associated with this UCB, then this member is used show/not show vehicle's polygon categories. The possible entries are FALSE and TRUE and possible categories are enumerated in the enum type **enCraft**, see the file "CodeBlockUser.h", i.e.:

```
vbShowComponent ()[craftGearFore] = FALSE;
```

double m_dViewPortFractionX

This member is used to set the fraction of the total window width that is used by AVDS to draw the scene. The rest of the window can be used for user defined graphics. This member ranges from 0.0 to 1.0.

double m_dViewPortFractionY

This member is used to set the fraction of the total window heigth that is used by AVDS to draw the scene. The rest of the window can be used for user defined graphics. This member ranges from 0.0 to 1.0.

G. UserCodeBlock Samples

General

There are four examples of the user of UCBs, *UserHUD*, *UserRadar*, *UserWingman*, and *UserText*. The source code for each of these example can be found in the directory “AVDS\model” in the subdirectories “UserHUD”, “UserRadar”, “UserWingman”, and “UserText”. respectively. *UserHUD* implements a heads-up display in the UCB. *UserRadar* implements a radar display in a UCB. *UserWingman* implements a second vehicle in a UCB. *UserText* implements the ability to add titles to AVDS.

Selecting/Configuring UserCodeBlocks

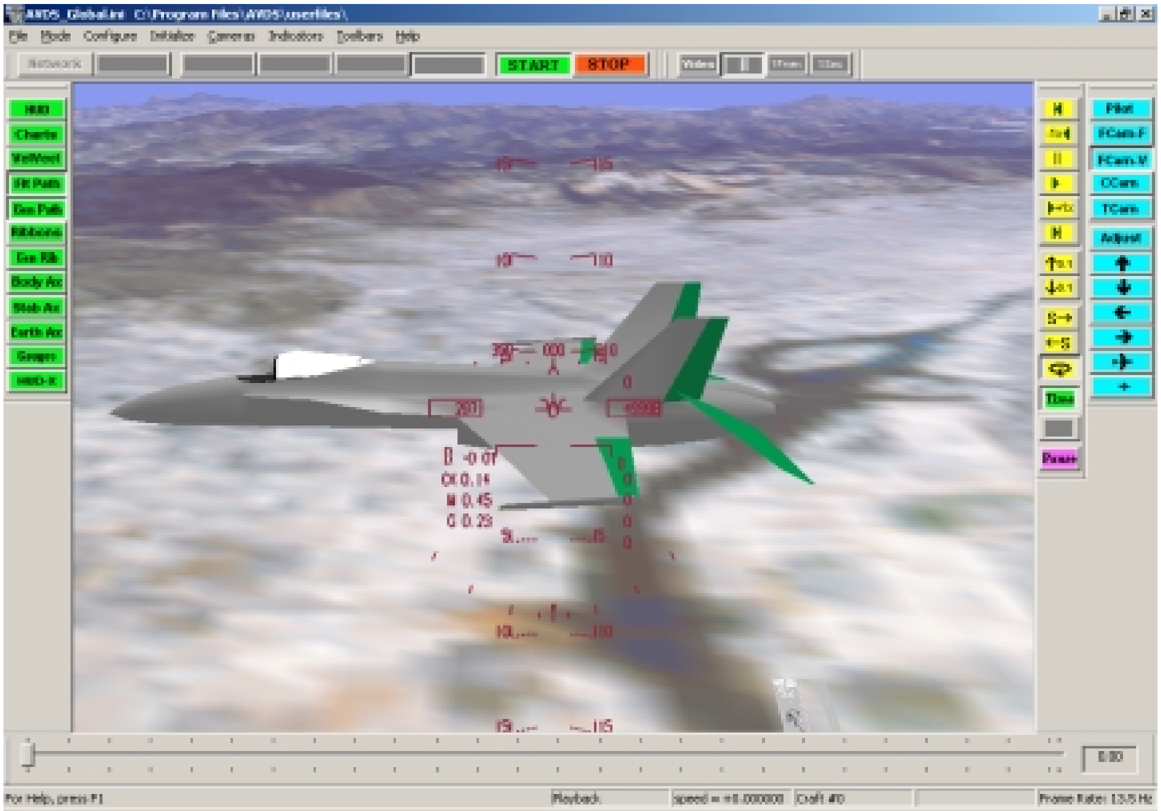
UCBs can be used in any of the following AVDS modes *Simulation*, *Model*, *Sim/Play*, and *Play*. To select/configure a UCB for a particular mode do the following:

1. select the mode
2. choose “Configure->Simulation” or “Configure->Playback Blocks” from the AVDS menu. (the available choice depends on the AVDS mode that was selected)
3. to configure a block ***right-click*** on it. This will bring up the block’s configuration dialog window, if one exists.
4. to select a block, click on the *square* next to it. This will cause icons of the block to appear in the Input/Output sections of the configuration dialog window. These inputs/outputs can be configured by dragging and dropping along with the rest of the input/outputs. Note: at this time, only one instance of a block can be used.

UserHUD

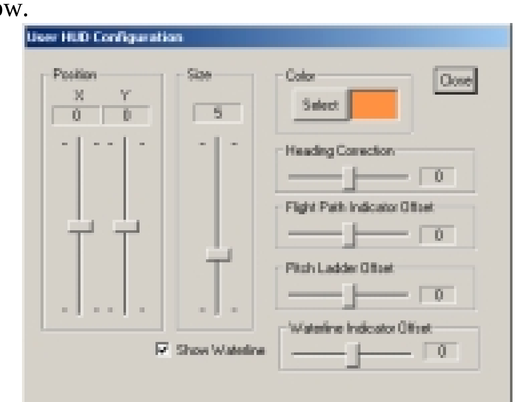
Description

The UserHUD UCB draws a heads-up display on the AVDS window. The inputs to the HUD are the inputs of the UserHUD block. These inputs are displayed digitally, used to build pitch/roll ladder, used to build the heading tape, and used to place the wind vector indicator.



Configuration

The UserHUD configuration is based on the AVDS internal HUD configuration. The HUD configuration dialog window is shown below.



Inputs

inputTime_sec - Time in seconds.
inputPhi_deg - Roll angle in degrees.
inputTheta_deg – Pitch angle in degrees.
inputPsi_deg - Heading in degrees.
inputV_ftpersec – Velocity in feet per second
inputAltitude_ft – Altitude in feet.
inputAltitudeDot_ftpersec **–** Rate of change ib altitude in feet per second.
inputAlpha_deg – Angle of attack in degrees.
inputBeta_deg **–** Angle of sideslip in degrees.
inputMach – Mach number.
inputGs – G loading.
inputUserData01 **–** User data to be placed in user data slot #1.
inputUserData02 **–** User data to be placed in user data slot #2.
inputUserData03 **–** User data to be placed in user data slot #3.
inputUserData04 **–** User data to be placed in user data slot #4.

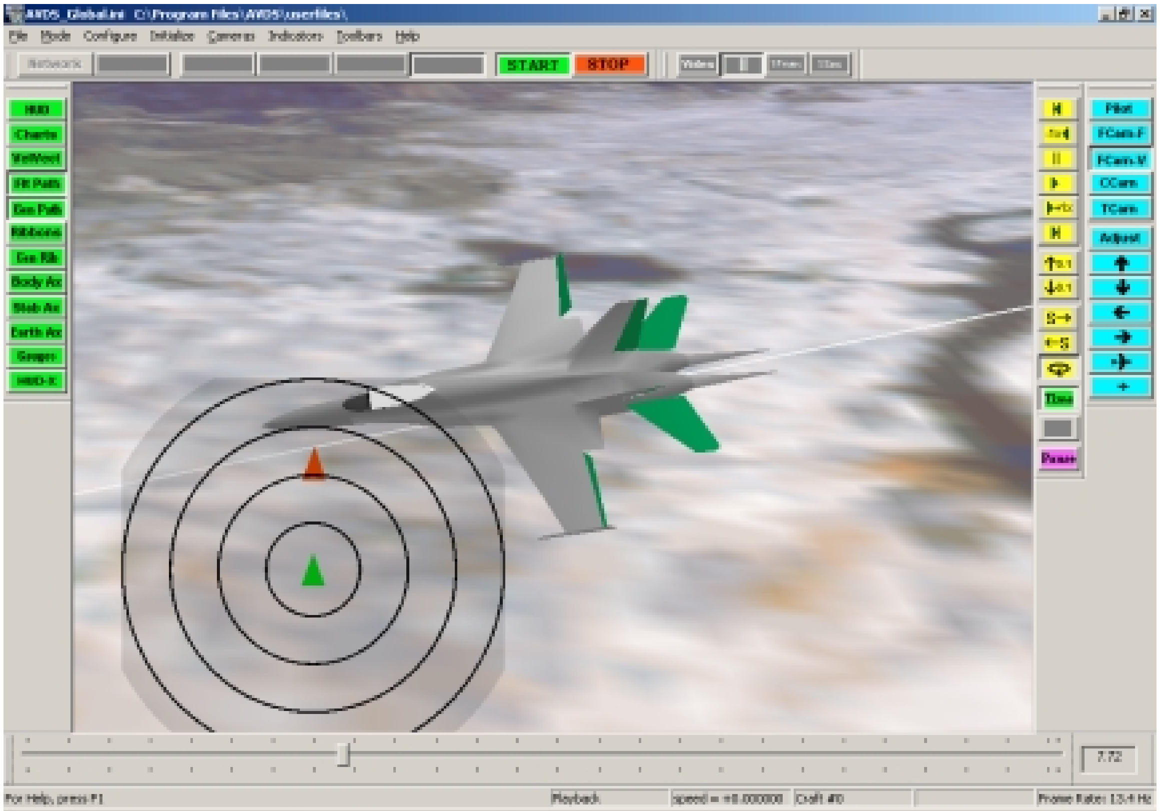
Outputs

none

UserRadar

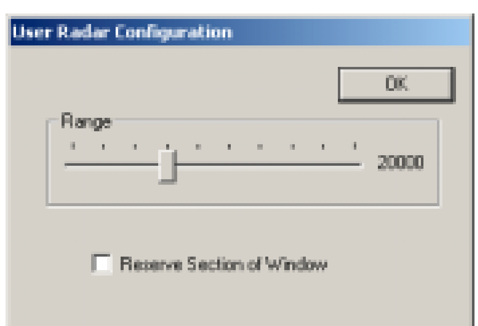
Description

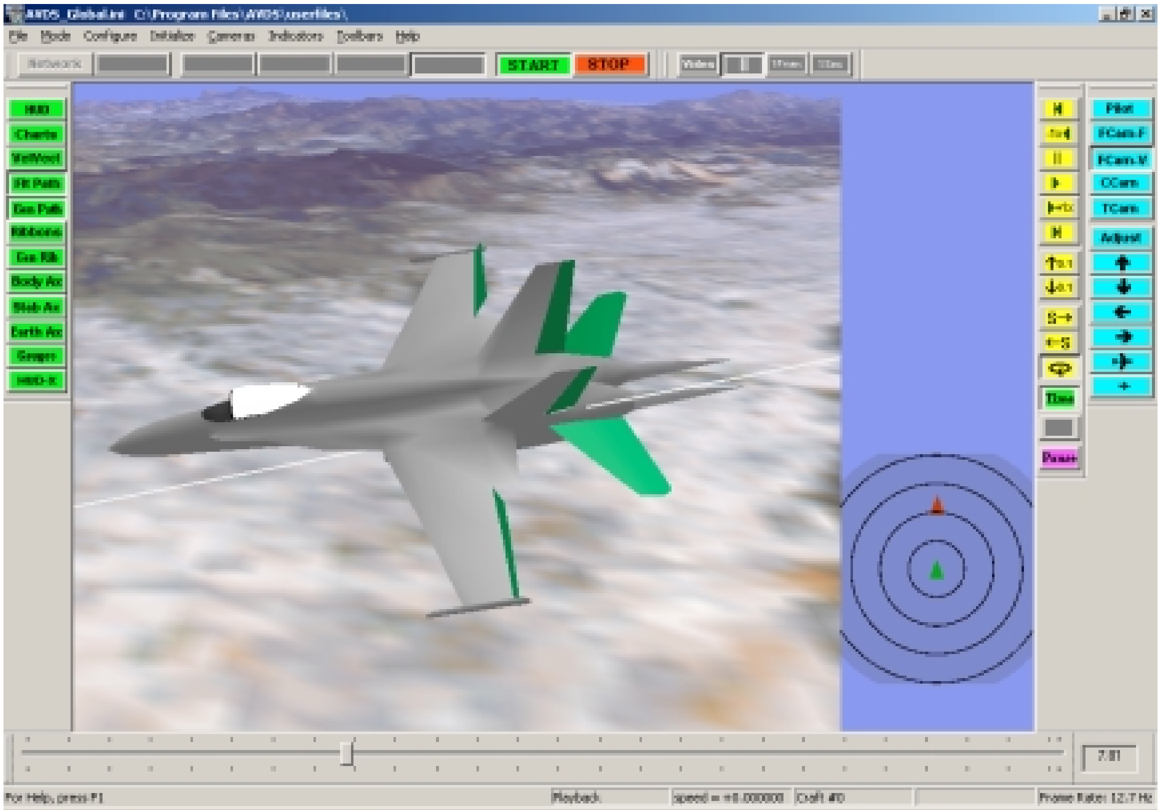
The UserRadar UCB generates a transparent display, similar to a radar display, which shows the relative positions of the vehicles. The positions of the vehicles are obtained from the inputs to the UCB. The range and display of the UserRadar UCB can be configured.



Configuration

Range **-** The UserRadar UCB range rings can be configured by moving the range slider pictured in the following figure.
Reserve Section of Window – Checking this box causes a portion of the AVDS window to be reserved for use of the radar display, as seen in the figure on the next page.





Inputs

> ***AC1_Heading_deg*** – Heading (degrees) of vehicle #1.
> ***AC1_North_ft*** – North position (feet) of vehicle #1.
> ***AC1_East_ft*** – East position (feet) of vehicle #1.
> ***AC2_Heading_deg*** – Heading (degrees) of vehicle #2.
> ***AC2_North_ft*** – North position (feet) of vehicle #2.
> ***AC2_East_ft*** – East position (feet) of vehicle #2.
> ***Note***: The positions are relative to the initial position.

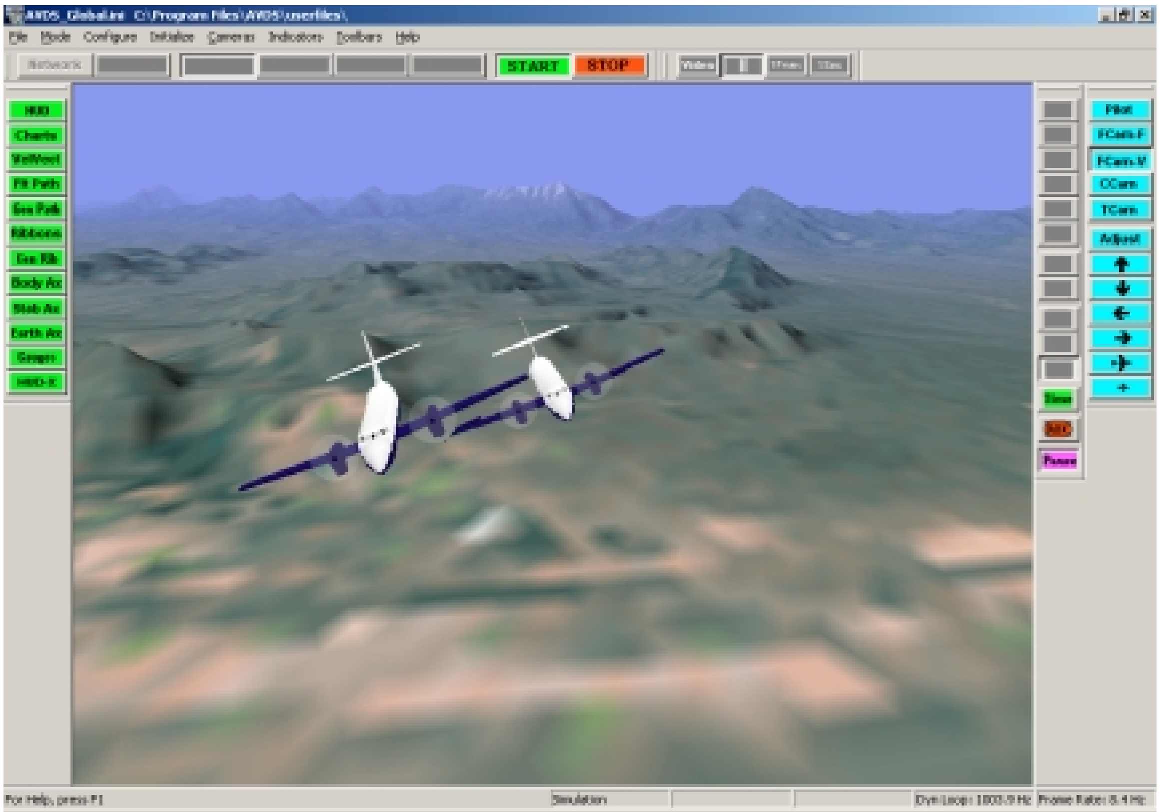
Outputs

> ***none***

UserWingman

Description

UserWingman creates a vehicle that "flies in formation" with the "lead" vehicle that is being flown in the AVDS simulation. This UCB demonstrates the ability to create a new AVDS entity and set its angular and linear positions in the scene. The UserWingman sample sets the craft type to 1, i.e. the second craft type in the craftcap.txt file. The inputs to the block, which are the coordinates of the lead aircraft, are modified and used to set the position of the wingman vehicle. This sample does not include a vehicle simulation, although it could, therefore the wingman vehicle's positions are directly proportional to the simulated vehicle's positions.



Configuration

none

Inputs

PosX_01 – North position (feet) of the lead aircraft.
PosY_01 – East position (feet) of the lead aircraft.
PosZ_01 – Altitude (feet) of the lead aircraft.
RotX_01 – Roll angle (deg) of the lead aircraft
RotY_01 – Pitch angle (deg) of the lead aircraft
RotZ_01 – Yaw angle (deg) of the lead aircraft
LatitudeStart – Starting latitude in degrees decimal, i.e. deg.dd.
LongitudeStart – Starting longitude in degrees decimal, i.e. deg.dd

Outputs

none

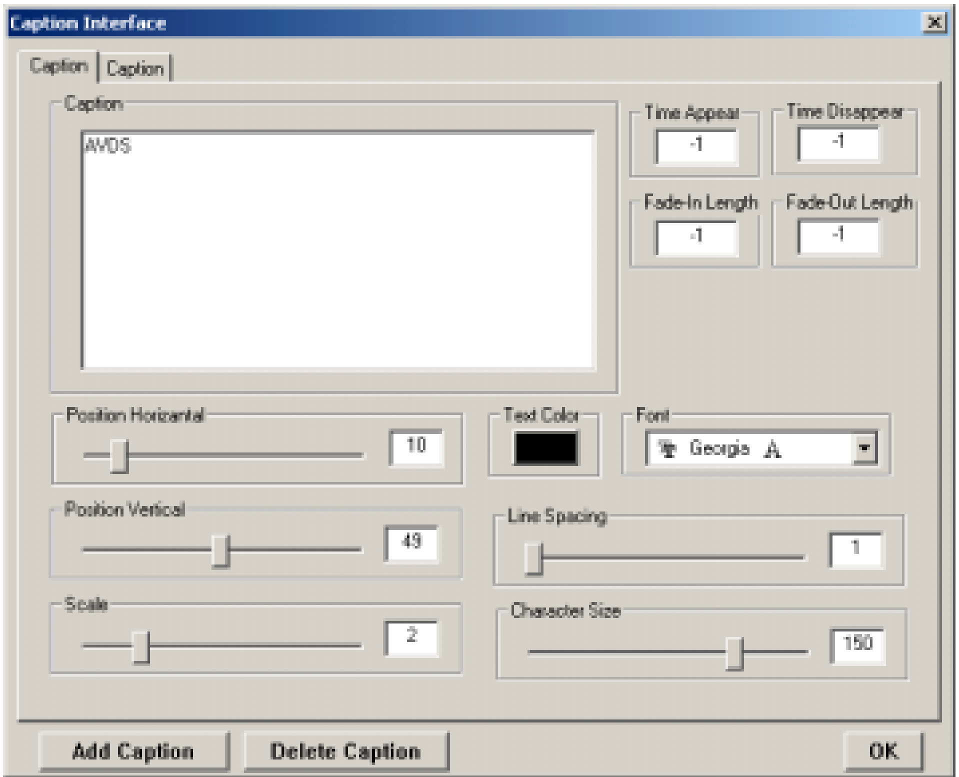
UserText

Description

The UserText UCB make it possible to add title and captions to AVDS during *Simulation*, *Model*, *Sim/Play*, and *Playback* modes.

Configuration

Add Caption – Inserts a new caption tab. This makes it possible to add text with different colors, sizes, positions, and fonts.
Delete Caption – Deletes the active caption tab.
Ok – Closes the configuration dialog window.
Caption – Area for entry of caption text. Can be entered with multiple lines.
Time Appear – The time (seconds) for the text to start to appear/fade-in.
Time Disappear – the time (seconds) for the text to completely disappear.
Fade-In Length – length of time (seconds) for the text to fade-in, starting at the ***Time Appear***.
Fade-Out Length – length of time (seconds) for the text to fade-out, ends at the ***Time Disappear***.
Position Horizontal – Horizontal position (percent) of the left edge of the text.
Position Vertical – Vertical position (percent) of the bottom of the first line of the text.
Scale – Scale factor multiplier for the text.
Text Color – This button brings up a color selection dialog window used to change the color of the text.
Font – This drop-down list displays the available fonts.
Line Spacing – Changes the spacing between the lines of text in the ***Caption*** area.
Character Size – Font size in points.



Inputs

Time – this is the time used to make captions appear and disappear. It is also used to cause the captions to fade in or fade out.

Outputs

none